Integer Set Coalescing for Polyhedral Compilation

Sven Verdoolaege

INRIA and KU Leuven

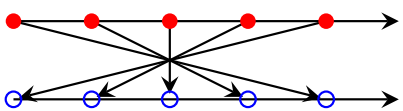
October 21, 2014

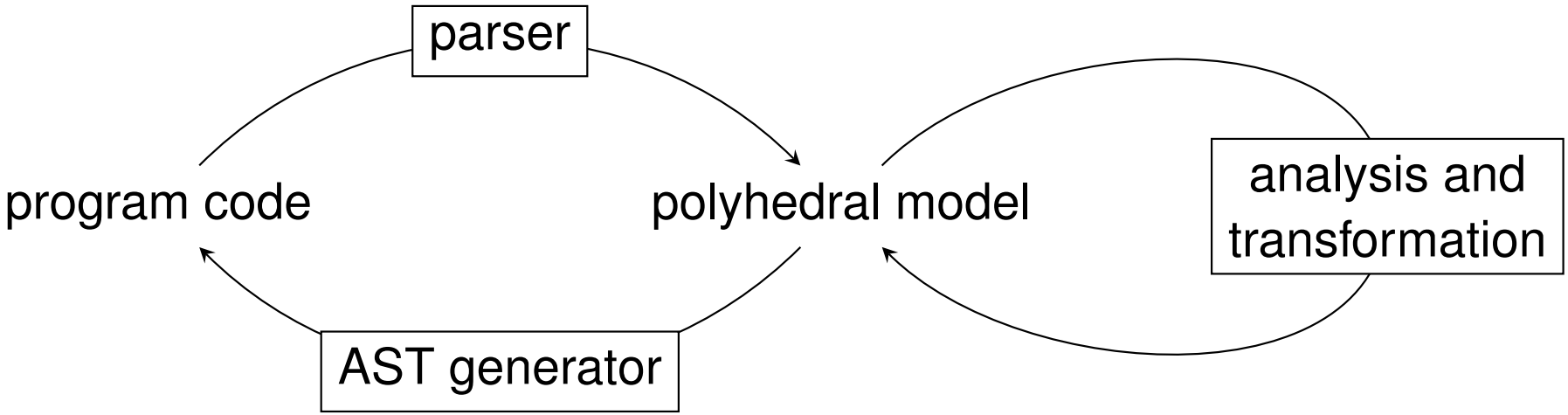
Outline

Outline

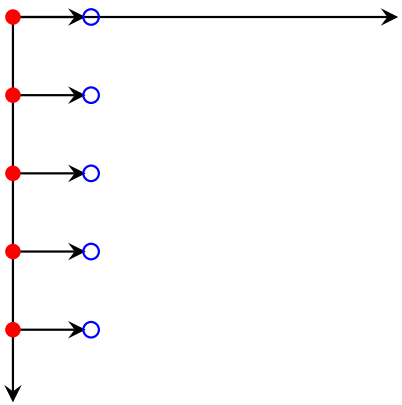
Polyhedral Compilation

```
for (i = 0; i <= N; ++i)
    a[i] = ...
for (i = 0; i <= N; ++i)
    b[i] = f(a[N-i])
```






```
for (i = 0; i <= N; ++i) {
    a[i] = ...
    b[N-i] = f(a[i])
}
```



Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(0,0); \mathtt{S}(0,1); \mathtt{S}(1,0); \mathtt{S}(1,1); \mathtt{T}(0); \mathtt{T}(1)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(0,0); \mathtt{S}(0,1); \mathtt{S}(1,0); \mathtt{S}(1,1); \mathtt{T}(0); \mathtt{T}(1)\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(0,0) \to \mathtt{A}(0); \mathtt{S}(0,1) \to \mathtt{A}(1); \mathtt{S}(1,0) \to \mathtt{A}(1); \mathtt{S}(1,1) \to \mathtt{A}(2)\}$$

$$R = \{\mathtt{R}() \to \mathtt{A}(2); \mathtt{T}(0) \to \mathtt{A}(0); \mathtt{T}(1) \to \mathtt{A}(1); \mathtt{T}(1) \to \mathtt{A}(0)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(0,0); \mathtt{S}(0,1); \mathtt{S}(1,0); \mathtt{S}(1,1); \mathtt{T}(0); \mathtt{T}(1)\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(0,0) \to \mathtt{A}(0); \mathtt{S}(0,1) \to \mathtt{A}(1); \mathtt{S}(1,0) \to \mathtt{A}(1); \mathtt{S}(1,1) \to \mathtt{A}(2)\}$$

$$R = \{\mathtt{R}() \to \mathtt{A}(2); \mathtt{T}(0) \to \mathtt{A}(0); \mathtt{T}(1) \to \mathtt{A}(1); \mathtt{T}(1) \to \mathtt{A}(0)\}$$

- Schedule (execution order)

$$\{\mathtt{R}() \to (0); \mathtt{S}(0,0) \to (1); \mathtt{S}(0,1) \to (2); \mathtt{S}(1,0) \to (3); \mathtt{S}(1,1) \to (4); \mathtt{T}(0) \to (5); \mathtt{T}(1) \to (6)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{ R(); S(0,0); S(0,1); S(1,0); S(1,1); T(0); T(1) \}$$

- Access relations (accessed array elements, W: write, R: read)

$$\{ R(); S(i,j) : 0 \le i < 2 \wedge 0 \le j < 2; T(k) : 0 \le k < 2; \}$$

$$W = \{ S(0,0) \to A(0); S(0,1) \to A(1); S(1,0) \to A(1); S(1,1) \to A(2) \}$$

$$R = \{ R() \to A(2); T(0) \to A(0); T(1) \to A(1); T(1) \to A(0) \}$$

- Schedule (execution order)

$$\{ R() \to (0); S(0,0) \to (1); S(0,1) \to (2); S(1,0) \to (3); S(1,1) \to (4); T(0) \to (5); T(1) \to (6) \}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{ R(); S(0,0); S(0,1); S(1,0); S(1,1); T(0); T(1) \}$$

- Access relations (accessed array elements)

$$\{ R(); S(i,j) : 0 \le i < 2 \wedge 0 \le j < 2; T(k) : 0 \le k < 2; \}$$

Constraints

- "read off", or
- obtained through abstract interpretation

$$W = \{ S(0,0) \to A(0); S(0,1) \to A(1); S(1,0) \to A(1); S(1,1) \to A(2) \}$$

$$R = \{ R() \to A(2); T(0) \to A(0); T(1) \to A(1); T(1) \to A(0) \}$$

- Schedule (execution order)

$$\{ R() \to (0); S(0,0) \to (1); S(0,1) \to (2); S(1,0) \to (3); S(1,1) \to (4); T(0) \to (5); T(1) \to (6) \}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(i,j) : 0 \leq i < 2 \wedge 0 \leq j < 2; \mathtt{T}(k) : 0 \leq k < 2\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(0,0) \to \mathtt{A}(0); \mathtt{S}(0,1) \to \mathtt{A}(1); \mathtt{S}(1,0) \to \mathtt{A}(1); \mathtt{S}(1,1) \to \mathtt{A}(2)\}$$

$$R = \{\mathtt{R}() \to \mathtt{A}(2); \mathtt{T}(0) \to \mathtt{A}(0); \mathtt{T}(1) \to \mathtt{A}(1); \mathtt{T}(1) \to \mathtt{A}(0)\}$$

- Schedule (execution order)

$$\{\mathtt{R}() \to (0); \mathtt{S}(0,0) \to (1); \mathtt{S}(0,1) \to (2); \mathtt{S}(1,0) \to (3); \mathtt{S}(1,1) \to (4); \mathtt{T}(0) \to (5); \mathtt{T}(1) \to (6)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(i,j) : 0 \le i < 2 \wedge 0 \le j < 2; \mathtt{T}(k) : 0 \le k < 2\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(0,0) \to \mathtt{A}(0); \mathtt{S}(0,1) \to \mathtt{A}(1); \mathtt{S}(1,0) \to \mathtt{A}(1); \mathtt{S}(1,1) \to \mathtt{A}(2)\}$$

$$R = \{\mathtt{R}() \to \mathtt{A}(0); \mathtt{S}(i,j) \to \mathtt{A}(i+j) : 0 \le i < 2 \wedge 0 \le j < 2; \mathtt{T}(k) \to \mathtt{A}(k) : 0 \le k < 2\}$$

- Schedule (execution order)

$$\{\mathtt{R}() \to (0); \mathtt{S}(0,0) \to (1); \mathtt{S}(0,1) \to (2); \mathtt{S}(1,0) \to (3); \mathtt{S}(1,1) \to (4); \mathtt{T}(0) \to (5); \mathtt{T}(1) \to (6)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(i,j) : 0 \le i < 2 \wedge 0 \le j < 2; \mathtt{T}(k) : 0 \le k < 2\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(i,j) \to \mathtt{A}(i+j) : 0 \le i < 2 \wedge 0 \le j < 2\}$$

$$R = \{\mathtt{R}() \to \mathtt{A}(2); \mathtt{T}(0) \to \mathtt{A}(0); \mathtt{T}(1) \to \mathtt{A}(1); \mathtt{T}(1) \to \mathtt{A}(0)\}$$

- Schedule (execution order)

$$\{\mathtt{R}() \to (0); \mathtt{S}(0,0) \to (1); \mathtt{S}(0,1) \to (2); \mathtt{S}(1,0) \to (3);$$
$$\mathtt{S}(1,1) \to (4); \mathtt{T}(0) \to (5); \mathtt{T}(1) \to (6)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(i,j) : 0 \le i < 2 \wedge 0 \le j < 2; \mathtt{T}(k) : 0 \le k < 2\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(i,j) \to \mathtt{A}(i+j) : 0 \le i < 2 \wedge 0 \le j < 2\}$$

$$R = \{\mathtt{R}() \to \mathtt{A}(2); \mathtt{T}(0) \to \mathtt{A}(0); \mathtt{T}(1) \to \mathtt{A}(1); \mathtt{T}(1) \to \mathtt{A}(0)\}$$

- Schedule (execution order)

$$\{\mathtt{R}() \to \mathtt{A}(2); \mathtt{T}(k) \to \mathtt{A}(0) : 0 \le k < 2; \mathtt{T}(k) \to \mathtt{A}(k) : 0 \le k < 2\}$$

$$\{\mathtt{R}() \to (0); \mathtt{S}(0,0) \to (1); \mathtt{S}(0,1) \to (2); \mathtt{S}(1,0) \to (3);$$
$$\mathtt{S}(1,1) \to (4); \mathtt{T}(0) \to (5); \mathtt{T}(1) \to (6)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(i,j) : 0 \le i < 2 \wedge 0 \le j < 2; \mathtt{T}(k) : 0 \le k < 2\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(i,j) \to \mathtt{A}(i+j) : 0 \le i < 2 \wedge 0 \le j < 2\}$$

$$R = \{\mathtt{R}() \to \mathtt{A}(2); \mathtt{T}(k) \to \mathtt{A}(0) : 0 \le k < 2; \mathtt{T}(k) \to \mathtt{A}(k) : 0 \le k < 2\}$$

- Schedule (execution order)

$$\{\mathtt{R}() \to (0); \mathtt{S}(0,0) \to (1); \mathtt{S}(0,1) \to (2); \mathtt{S}(1,0) \to (3);$$
$$\mathtt{S}(1,1) \to (4); \mathtt{T}(0) \to (5); \mathtt{T}(1) \to (6)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(i,j) : 0 \le i < 2 \wedge 0 \le j < 2; \mathtt{T}(k) : 0 \le k < 2\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(i,j) \to \mathtt{A}(i+j) : 0 \le i < 2 \wedge 0 \le j < 2\}$$

$$\{\mathtt{R}() \to (0); \mathtt{S}(i,j) \to (1+2i+j) : 0 \le i,j < 2; \mathtt{T}(k) \to (5+k) : 0 \le k < 2\}$$

- Schedule (execution order)

$$\{\mathtt{R}() \to (0); \mathtt{S}(0,0) \to (1); \mathtt{S}(0,1) \to (2); \mathtt{S}(1,0) \to (3); \mathtt{S}(1,1) \to (4); \mathtt{T}(0) \to (5); \mathtt{T}(1) \to (6)\}$$

Polyhedral Model

```
R:   h(A[2]);
     for (int i = 0; i < 2; ++i)
         for (int j = 0; j < 2; ++j)
S:           A[i + j] = f(i, j);
     for (int k = 0; k < 2; ++k)
T:           g(A[k], A[0]);
```

- instance based
- compact representation

- Instance set (set of statement instances)

$$I = \{\mathtt{R}(); \mathtt{S}(i,j) : 0 \le i < 2 \wedge 0 \le j < 2; \mathtt{T}(k) : 0 \le k < 2\}$$

- Access relations (accessed array elements; W: write, R: read)

$$W = \{\mathtt{S}(i,j) \to \mathtt{A}(i+j) : 0 \le i < 2 \wedge 0 \le j < 2\}$$

- Schedule (execution order)

$$\{\mathtt{R}() \to (0); \mathtt{S}(i,j) \to (1+2i+j) : 0 \le i,j < 2; \mathtt{T}(k) \to (5+k) : 0 \le k < 2\}$$
$$\{\mathtt{R}() \to (0,0,0); \mathtt{S}(i,j) \to (1,i,j) : 0 \le i,j < 2; \mathtt{T}(k) \to (2,k,0) : 0 \le k < 2\}$$

$$\{\mathtt{R}() \to (0); \mathtt{S}(0,0) \to (1); \mathtt{S}(0,1) \to (2); \mathtt{S}(1,0) \to (3); \mathtt{S}(1,1) \to (4); \mathtt{T}(0) \to (5); \mathtt{T}(1) \to (6)\}$$

Equivalent Representations

extensive $\{ S(0,0); S(0,1); S(1,0); S(1,1) \}$

$= \{ S(i,j) : (i = 0 \wedge j = 0) \vee (i = 0 \wedge j = 1) \vee (i = 1 \wedge j = 0) \vee (i = 1 \wedge j = 1) \}$

intensive $\{ S(i,j) : 0 \leq i < 2 \wedge 0 \leq j < 2 \}$

Equivalent Representations

extensive $\{ S(0,0); S(0,1); S(1,0); S(1,1) \}$

$= \{ S(i,j) : (i = 0 \wedge j = 0) \vee (i = 0 \wedge j = 1) \vee (i = 1 \wedge j = 0) \vee (i = 1 \wedge j = 1) \}$

intensive $\{ S(i,j) : 0 \leq i < 2 \wedge 0 \leq j < 2 \}$

alternative $\{ S(i,j) : (i = 0 \wedge 0 \leq j < 2) \vee (i = 1 \wedge 0 \leq j < 2) \}$

Equivalent Representations

extensive	$\{ S(0,0); S(0,1); S(1,0); S(1,1) \}$	4
	$= \{ S(i,j) : (i = 0 \wedge j = 0) \vee (i = 0 \wedge j = 1) \vee$	
	$(i = 1 \wedge j = 0) \vee (i = 1 \wedge j = 1) \}$	
intensive	$\{ S(i,j) : 0 \leq i < 2 \wedge 0 \leq j < 2 \}$	1
alternative	$\{ S(i,j) : (i = 0 \wedge 0 \leq j < 2) \vee (i = 1 \wedge 0 \leq j < 2) \}$	2

In general, representation with fewer disjuncts is preferred

- (usually) occupies less memory
- operations can be performed more efficiently
- the outcome of some operations depends on chosen representation
  - transitive closure approximation
  - AST generation

$\Rightarrow$ coalescing: replace representation by one with fewer disjuncts

Effect on AST Generation — guide

Without coalescing

```
for (int c0 = 1; c0 <= min(2 * M, N); c0 += 1)
  S1(c0);
for (int c0 = max(1, 2 * M + 1); c0 <= N; c0 += 1)
  S1(c0);
for (int c0 = N + 1; c0 <= 2 * N; c0 += 1)
  S2(c0);
```

With coalescing

```
for (int c0 = 1; c0 <= N; c0 += 1)
  S1(c0);
for (int c0 = N + 1; c0 <= 2 * N; c0 += 1)
  S2(c0);
```

Effect on AST Generation — cholesky2

Without coalescing (fragment)

```
for (int c0 = 1; c0 < 3 * M - 1; c0 += 3) {
  S3((c0 + 2) / 3);
  if (3 * M >= c0 + 8) {
    for (int c1 = (c0 + 5) / 3; c1 <= M; c1 += 1) {
      S6((c0 + 2) / 3, c1);
      for (int c4 = (c0 + 5) / 3; c4 < c1; c4 += 1)
        S5(c4, c1, (c0 + 2) / 3);
    }
  } else if (c0 + 5 == 3 * M)
    S6(M - 1, M);
}
```

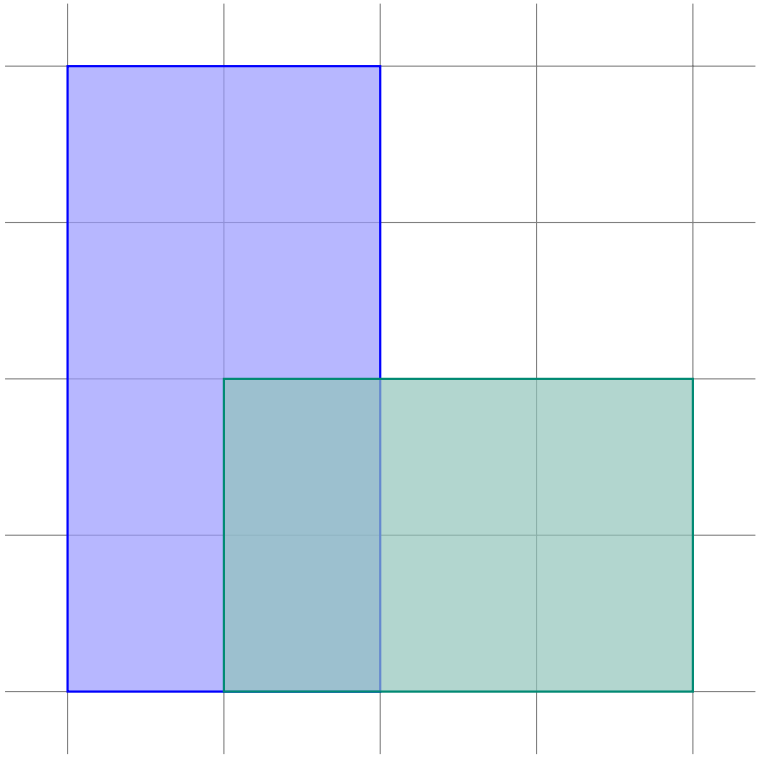
With coalescing (fragment)

```
for (int c0 = 1; c0 < 3 * M - 1; c0 += 3) {
  S3((c0 + 2) / 3);
  for (int c1 = (c0 + 5) / 3; c1 <= M; c1 += 1) {
    S6((c0 + 2) / 3, c1);
    for (int c4 = (c0 + 5) / 3; c4 < c1; c4 += 1)
      S5(c4, c1, (c0 + 2) / 3);
  }
}
```

Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

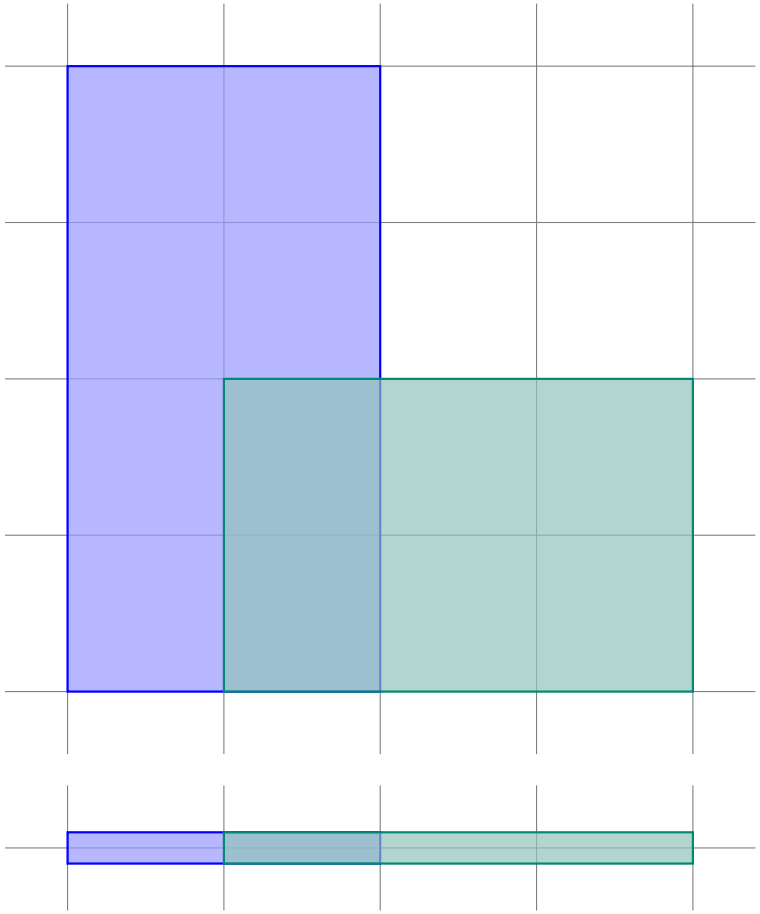
- Projection



Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

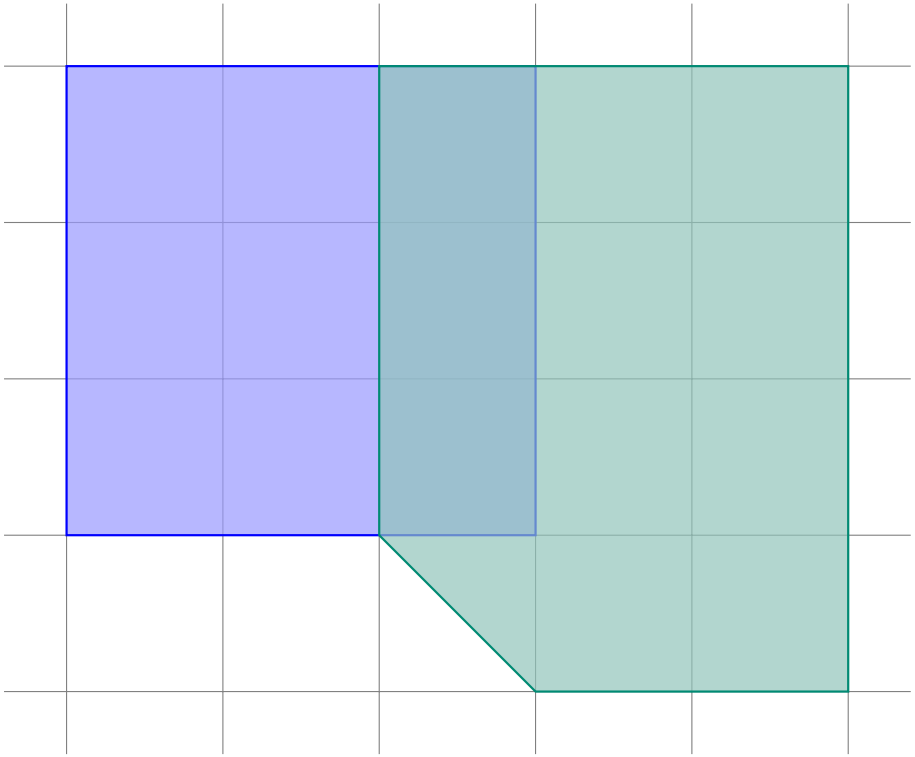
- Projection



Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

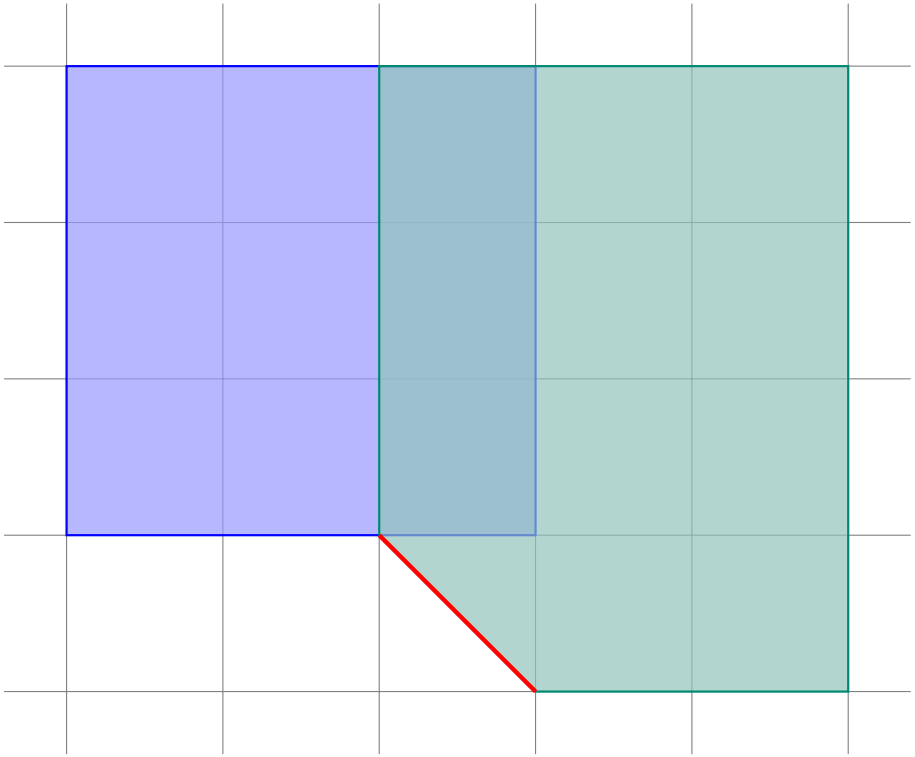
- Projection
- Subtraction



Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

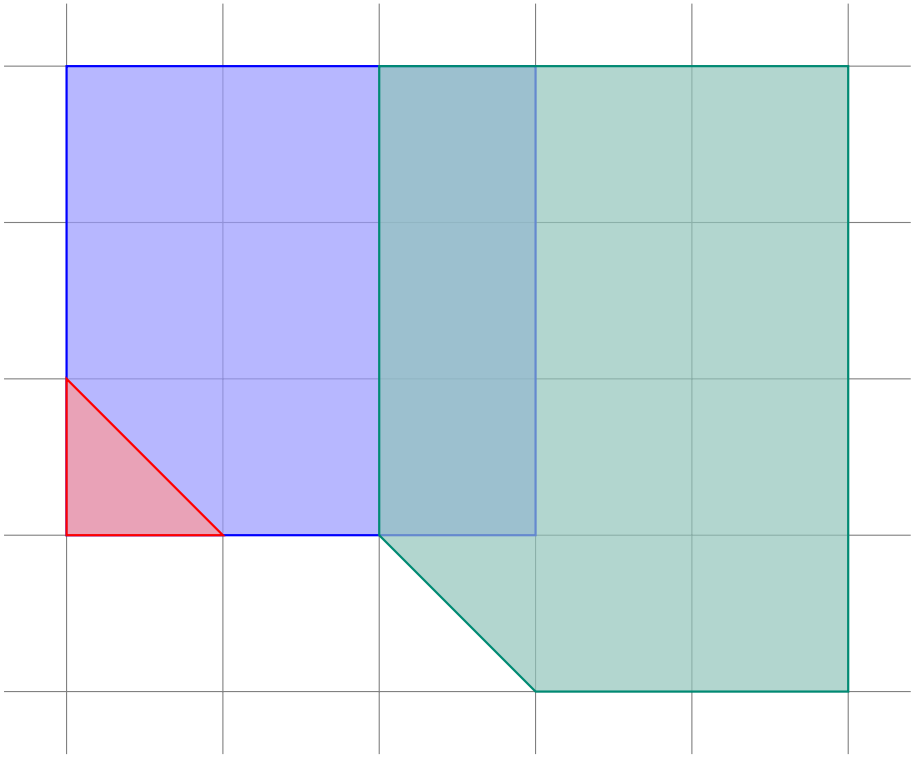
- Projection
- Subtraction



Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

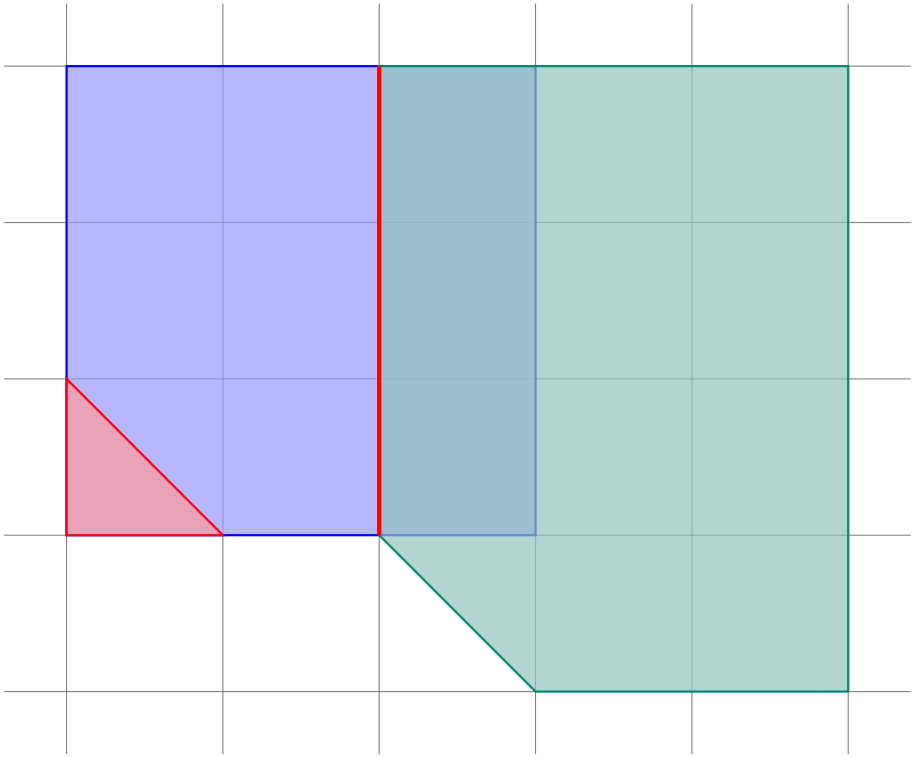
- Projection
- Subtraction



Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

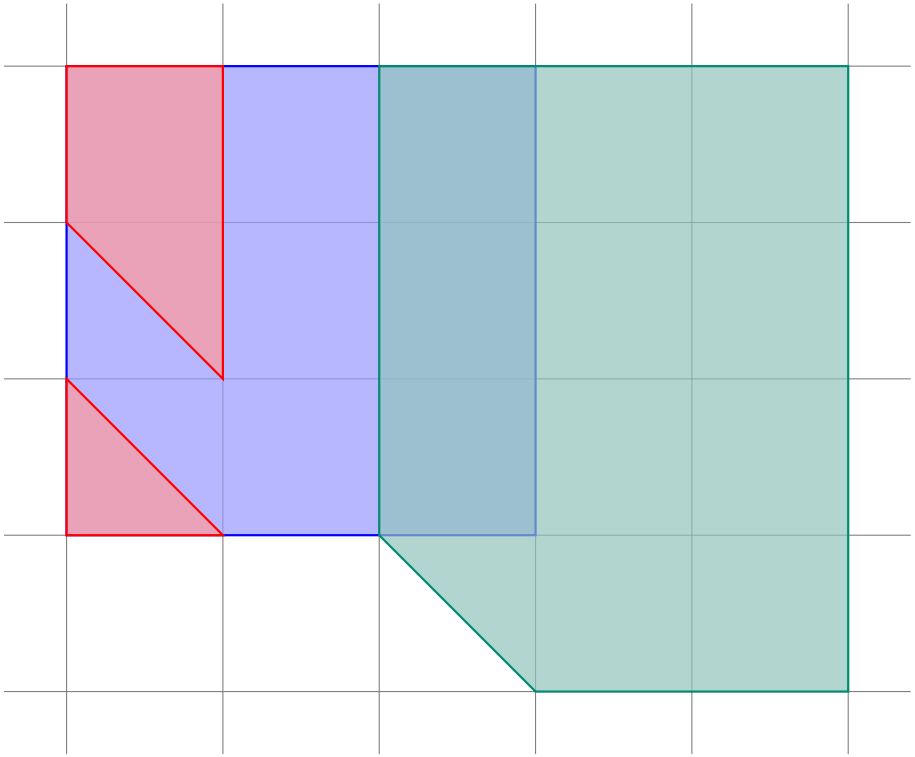
- Projection
- Subtraction



Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction



Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

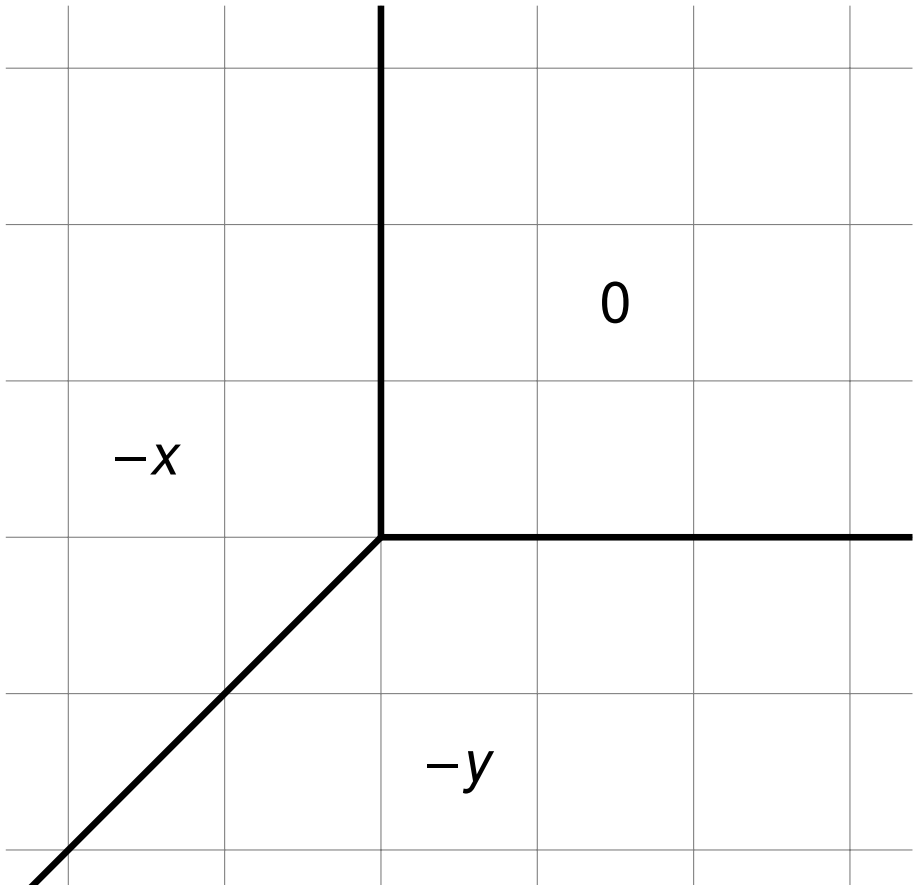
$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$

Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$

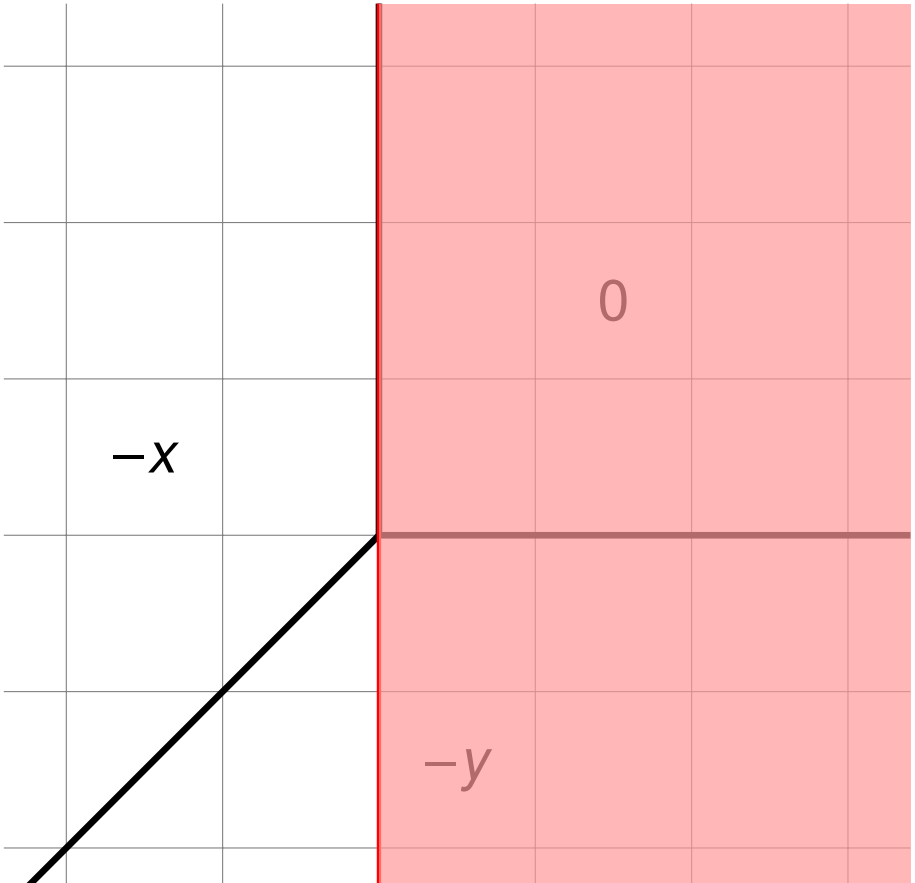


Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$

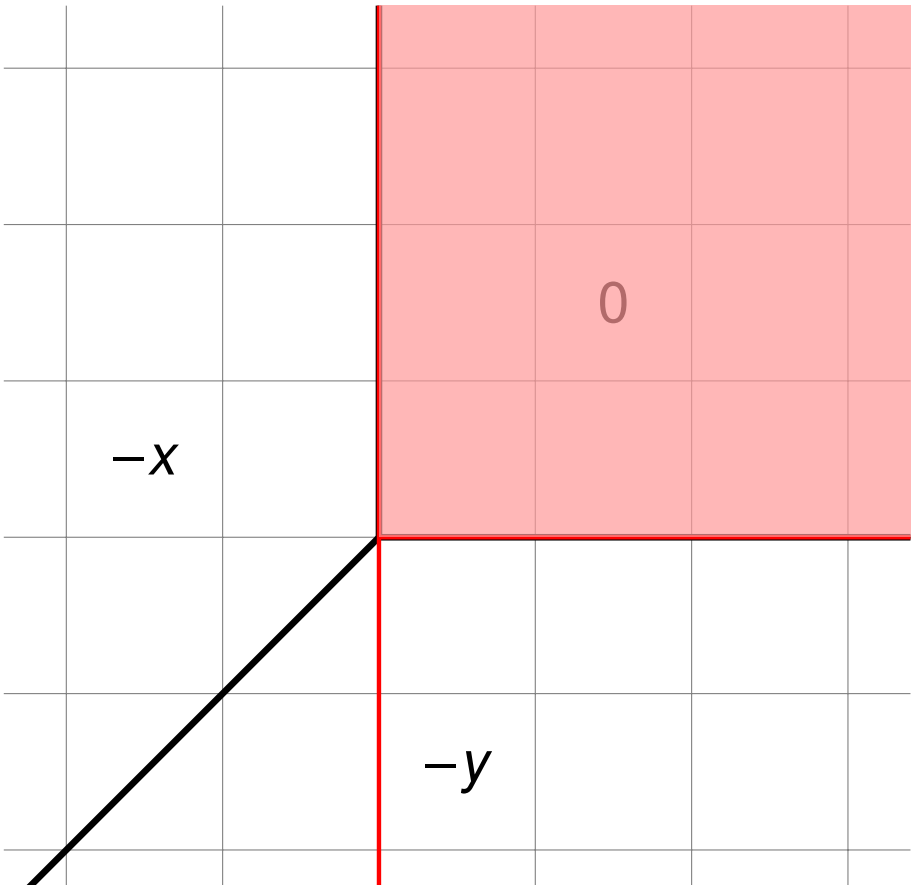


Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$

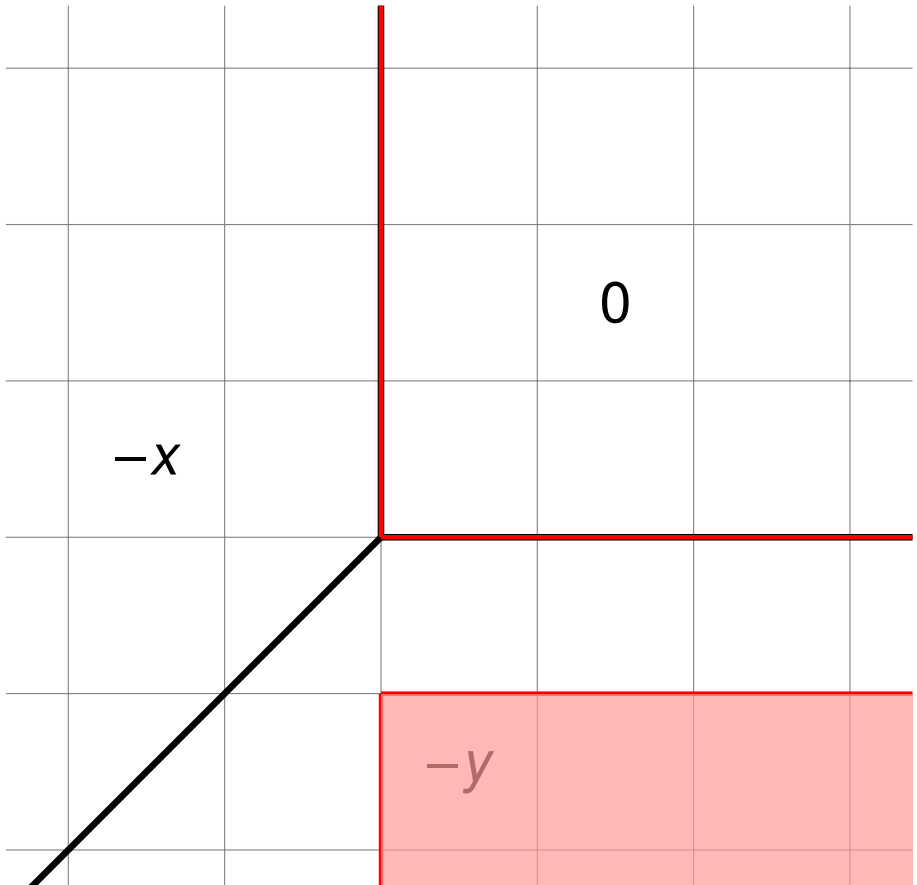


Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$

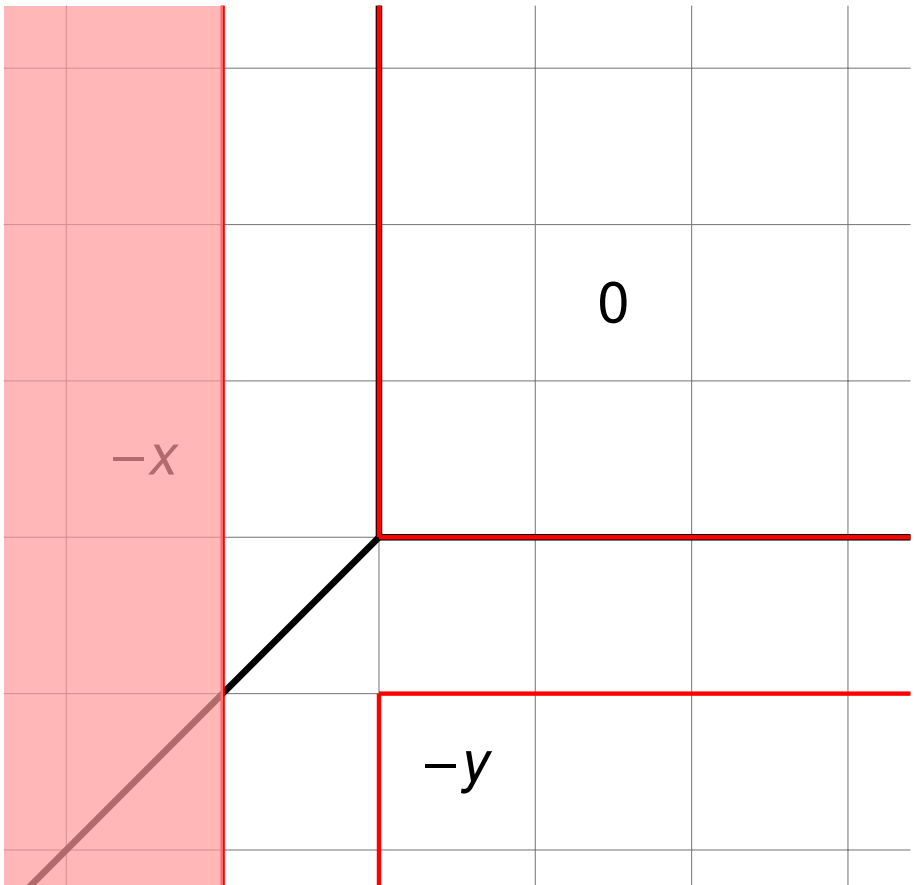


Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$

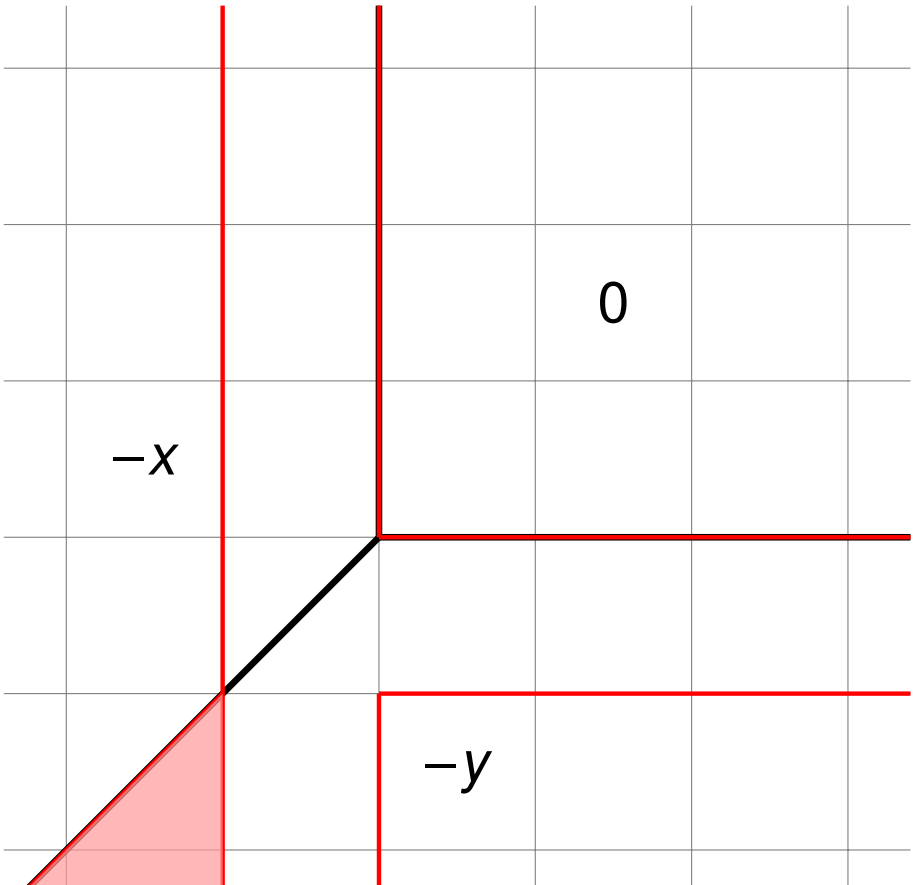


Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$

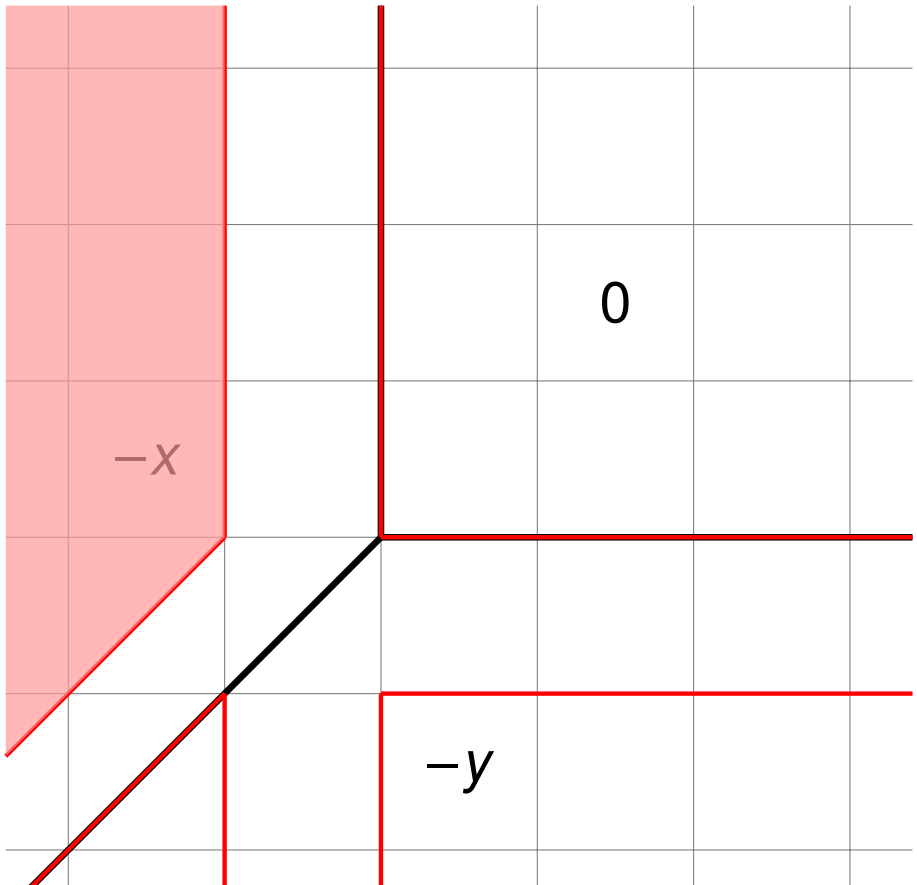


Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$

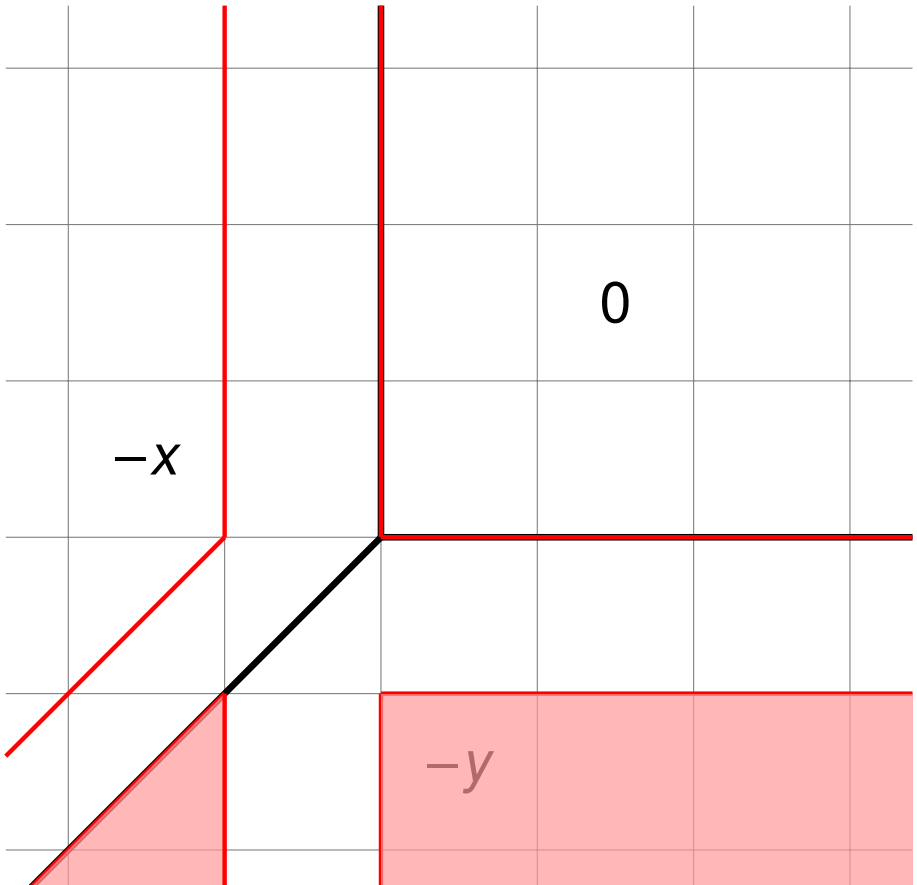


Causes of Splintering

Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming

$$\min\{(x, y) \to (z) : z \geq 0 \wedge x + z \geq 0 \wedge x + y \geq 0\}$$



Causes of Splintering

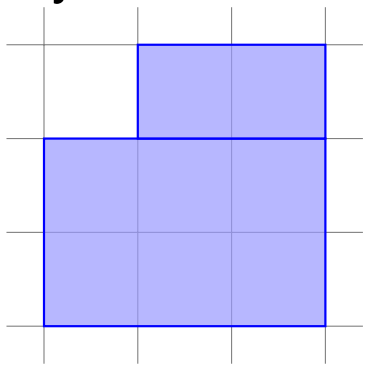
Several operations on integer sets may introduce coalescing opportunities

- Projection
- Subtraction
- Parametric integer programming
- Union

Traditional “Coalescing”

Traditional method (e.g., in CLooG with original PolyLib backend)

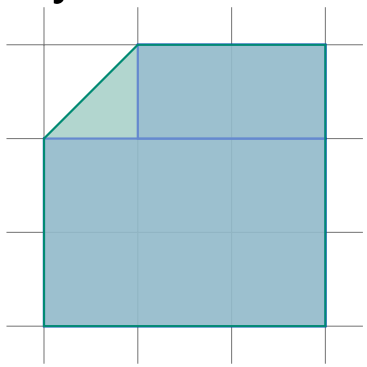
1. Compute convex hull H of S
2. Remove integer elements not in S from H
   $\Rightarrow H \setminus (H \setminus S)$



Traditional “Coalescing”

Traditional method (e.g., in CLooG with original PolyLib backend)

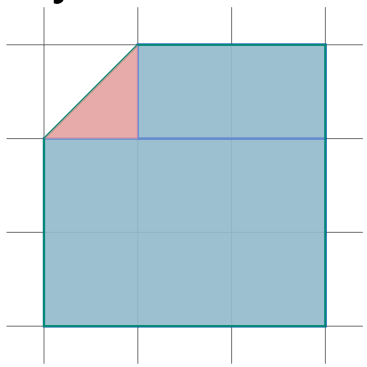
1. Compute convex hull H of S
2. Remove integer elements not in S from H
   $\Rightarrow H \setminus (H \setminus S)$



Traditional "Coalescing"

Traditional method (e.g., in CLooG with original PolyLib backend)

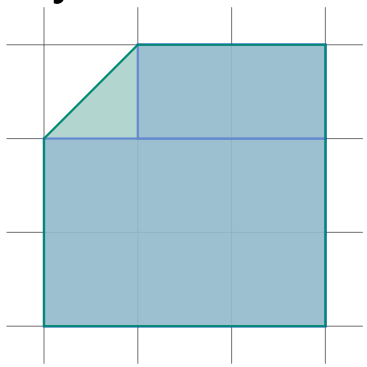
1. Compute convex hull H of S
2. Remove integer elements not in S from H
   $\Rightarrow H \setminus (H \setminus S)$



Traditional “Coalescing”

Traditional method (e.g., in CLooG with original PolyLib backend)

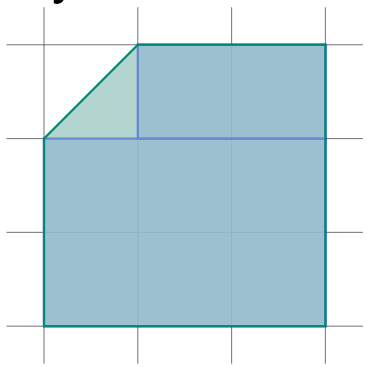
1. Compute convex hull H of S
2. Remove integer elements not in S from H
   $\Rightarrow H \setminus (H \setminus S)$



Traditional "Coalescing"

Traditional method (e.g., in CLooG with original PolyLib backend)

1. Compute convex hull H of S
2. Remove integer elements not in S from H
   $\Rightarrow H \setminus (H \setminus S)$



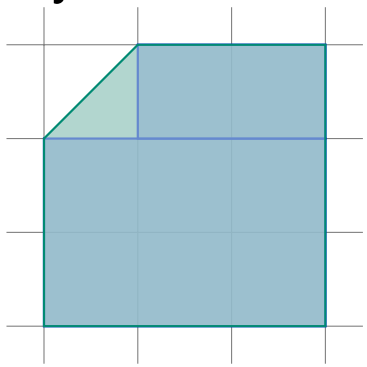
Issues:

- Convex hull may have exponential number of constraints
  We may be able to remove some of them, but we still need to compute them first.
- Constraints of convex hull may have very large coefficients
- Convex hull is an operation on *rational* sets
  - $\Rightarrow$ mixture of operation on rational sets (convex hull) and integer sets (set subtraction)
  - $\Rightarrow$ in `isl`, convex hull operation not fully defined on sets with existentially quantified variables
- Convex hull is costly to compute

Traditional "Coalescing"

Traditional method (e.g., in CLooG with original PolyLib backend)

1. Compute convex hull H of S
2. Remove integer elements not in S from H
   $\Rightarrow H \setminus (H \setminus S)$



Issues:

- Convex hull may have exponential number of constraints
  We may be able to remove some of them, but we still need to compute them first.
- Constraints of convex hull may have very large coefficients
- Convex hull is an operation on *rational* sets
  - $\Rightarrow$ mixture of operation on rational sets (convex hull) and integer sets (set subtraction)
  - $\Rightarrow$ in `isl`, convex hull operation not fully defined on sets with existentially quantified variables
- Convex hull is costly to compute

Effect on AST Generation — covariance

With isl coalescing (in this case same result as no coalescing)

```
for (long c1 = n >= 1 ? ((n - 1) % 32) - n - 31 : 0;
     c1 <= (n >= 1 ? n - 1 : 0); c1 += 32) {
  /* .. */
}
```

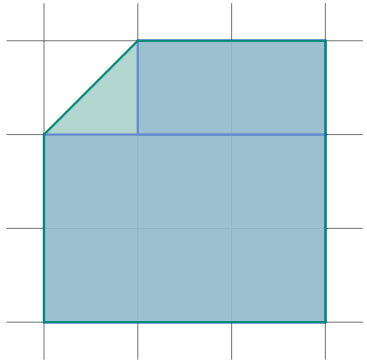
With convex hull based "coalescing"

```
for (long c1 = 32 * floord(-1073741839 * n -
    32749125633, 68719476720) - 1073741792; c1 <=
    floord(715827882 * n + 357913941, 1431655765) +
    1073741823; c1 += 32) {
  /* .. */
}
```

Traditional "Coalescing"

Traditional method (e.g., in CLooG with original PolyLib backend)

1. Compute convex hull H of S
2. Remove integer elements not in S from H
   $\Rightarrow H \setminus (H \setminus S)$



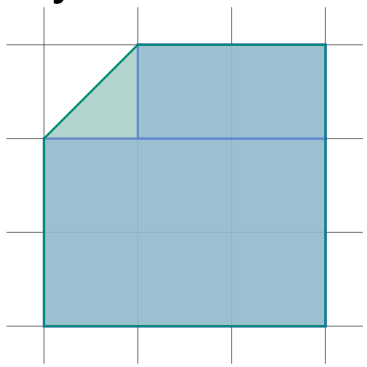
Issues:

- Convex hull may have exponential number of constraints
  We may be able to remove some of them, but we still need to compute them first.
- Constraints of convex hull may have very large coefficients
- Convex hull is an operation on *rational* sets
  - $\Rightarrow$ mixture of operation on rational sets (convex hull) and integer sets (set subtraction)
  - $\Rightarrow$ in `isl`, convex hull operation not fully defined on sets with existentially quantified variables
- Convex hull is costly to compute

Traditional "Coalescing"

Traditional method (e.g., in CLooG with original PolyLib backend)

1. Compute convex hull H of S
2. Remove integer elements not in S from H
   $\Rightarrow H \setminus (H \setminus S)$



Issues:

- Convex hull may have exponential number of constraints
  We may be able to remove some of them, but we still need to compute them first.
- Constraints of convex hull may have very large coefficients
- Convex hull is an operation on *rational* sets
  - $\Rightarrow$ mixture of operation on rational sets (convex hull) and integer sets (set subtraction)
  - $\Rightarrow$ in `isl`, convex hull operation not fully defined on sets with existentially quantified variables
- Convex hull is costly to compute

AST Generation Times

Generation times on `isl` AST generation test cases

`isl` coalescing	16.0s
no coalescing	16.3s
convex hull (FM)	24m00s
convex hull (wrapping)	6m40s

Note: `isl` may not have the most efficient convex hull implementation

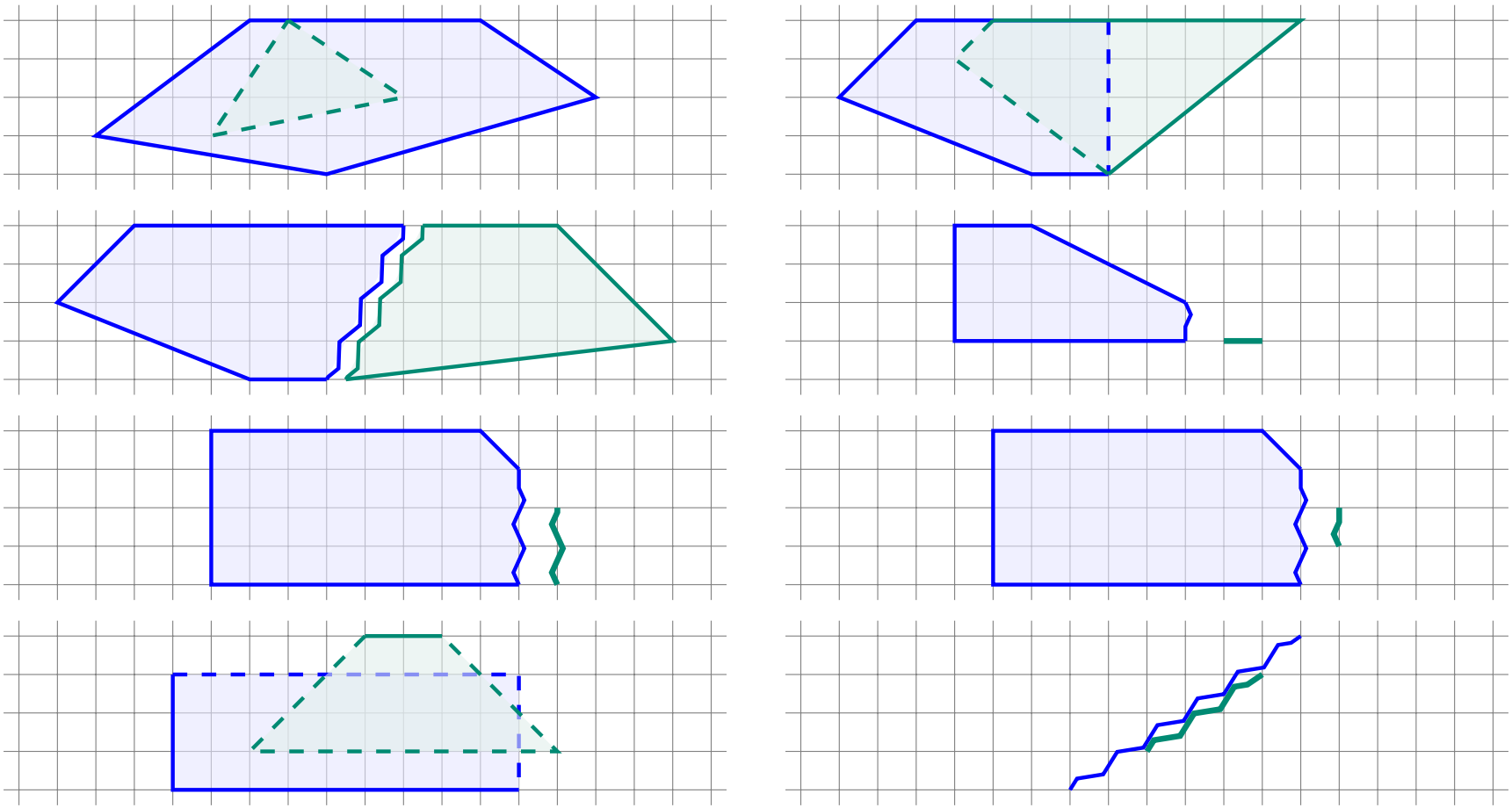
However, double description based implementations are costly too

Outline

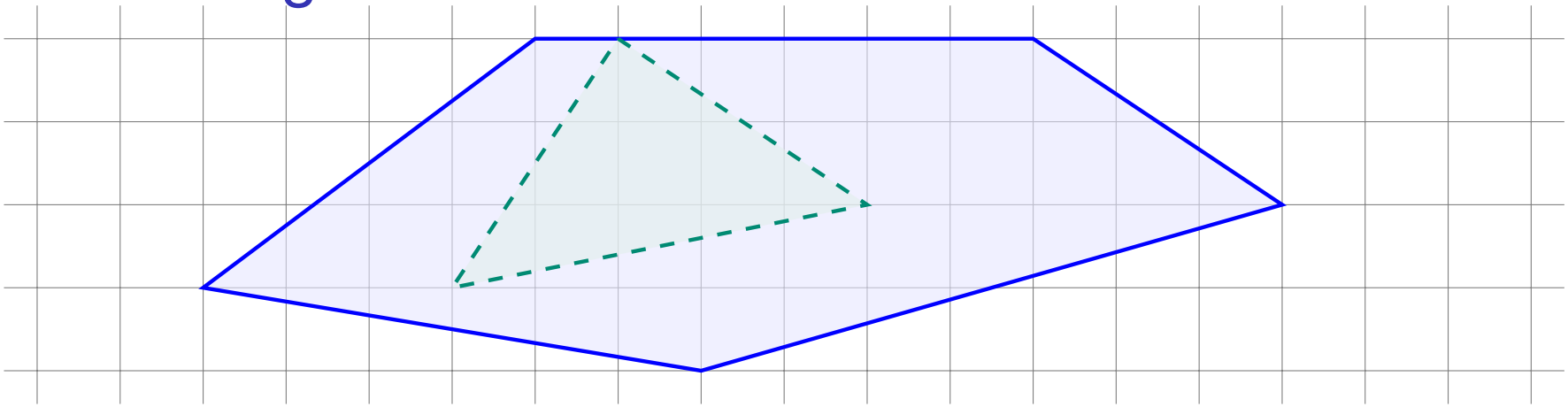
Coalescing in isl

Coalescing in isl

- never increases the total number of constraints
- based on solving LP problems with same dimension as input set
- recognizes a set of patterns



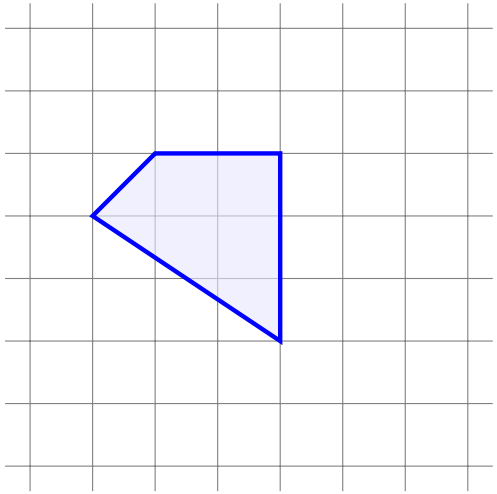
Coalescing Cases



Constraint types

Given two disjuncts A and B

For each affine constraint $t(\mathbf{x}) \geq 0$ of A, determine its effect on B



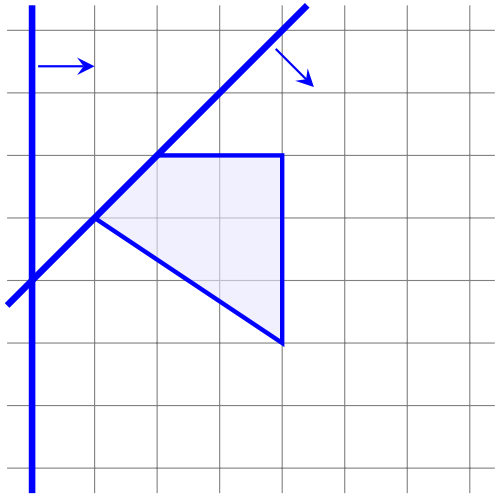
Note: affine expression $t(\mathbf{x}) \geq 0$ has integer coefficients
min and max computed using (incremental) LP solver

Constraint types

Given two disjuncts A and B

For each affine constraint $t(\mathbf{x}) \geq 0$ of A, determine its effect on B

- $\min t(\mathbf{x}) > -1$ over B
  $\Rightarrow$valid constraint



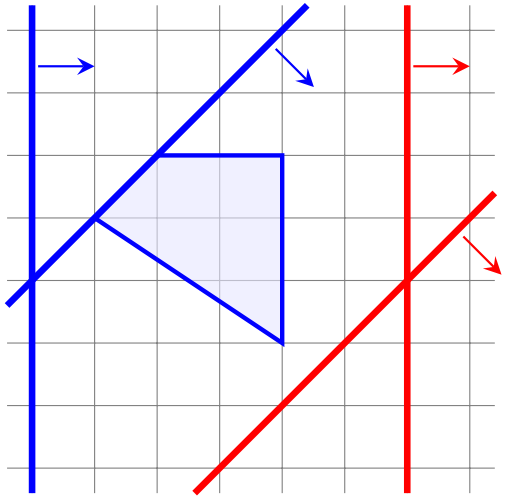
Note: affine expression $t(\mathbf{x}) \geq 0$ has integer coefficients
min and max computed using (incremental) LP solver

Constraint types

Given two disjuncts A and B

For each affine constraint $t(\mathbf{x}) \geq 0$ of A, determine its effect on B

- $\min t(\mathbf{x}) > -1$ over B
  $\Rightarrow$valid constraint
- $\max t(\mathbf{x}) < 0$ over B
  $\Rightarrow$separating constraint



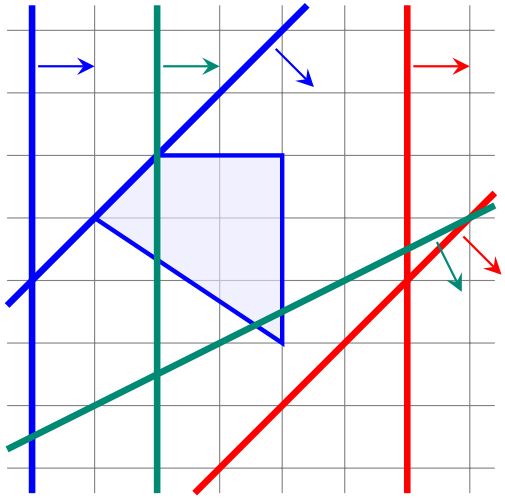
Note: affine expression $t(\mathbf{x}) \geq 0$ has integer coefficients
min and max computed using (incremental) LP solver

Constraint types

Given two disjuncts A and B

For each affine constraint $t(\mathbf{x}) \geq 0$ of A, determine its effect on B

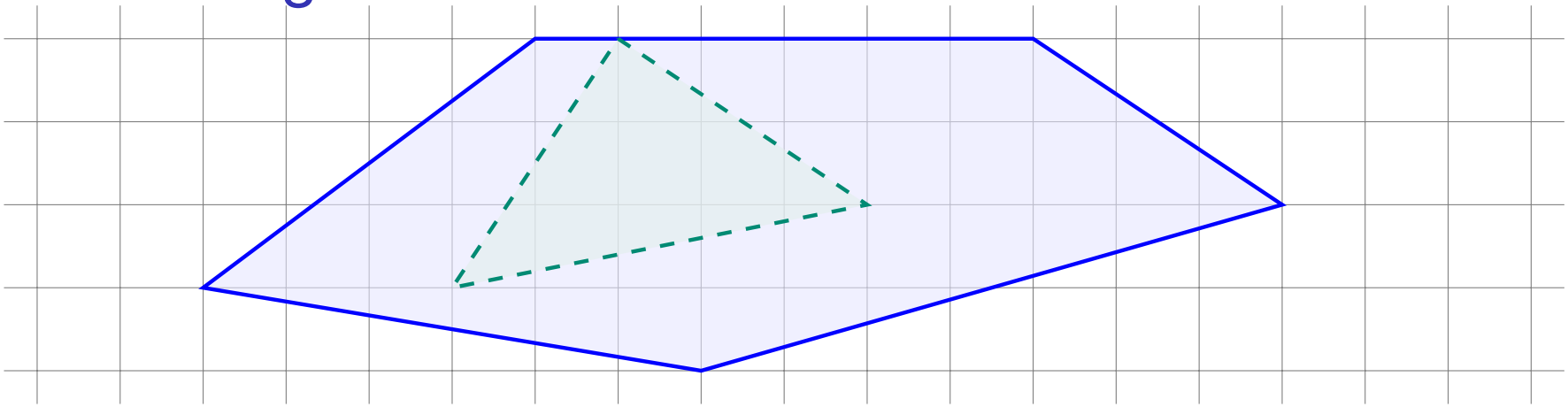
- $\min t(\mathbf{x}) > -1$ over B
  $\Rightarrow$valid constraint
- $\max t(\mathbf{x}) < 0$ over B
  $\Rightarrow$separating constraint



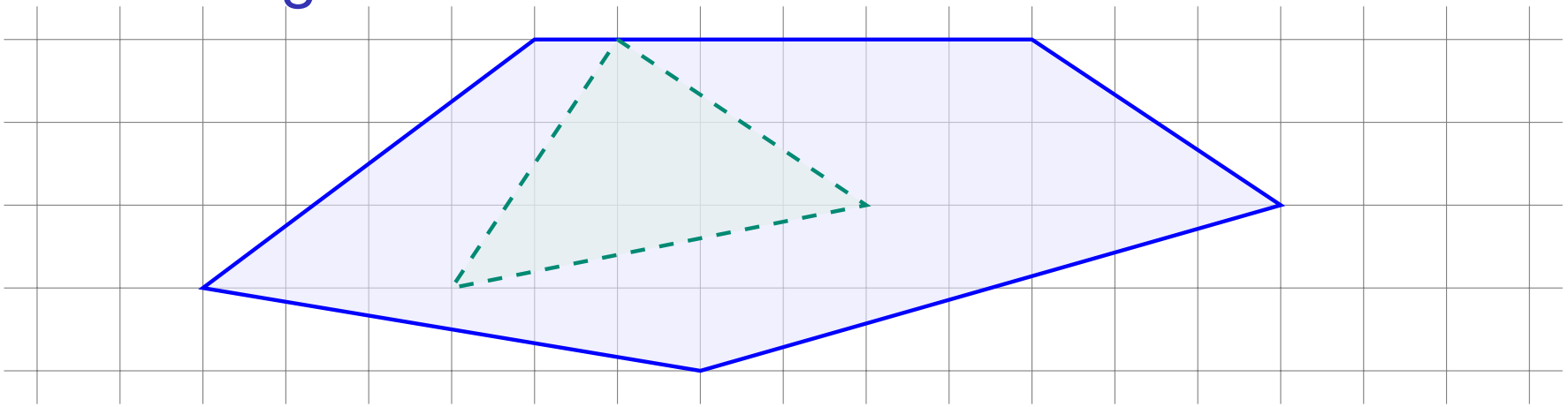
- otherwise (attains both positive and negative values over B)
  $\Rightarrow$cut constraint

Note: affine expression $t(\mathbf{x}) \geq 0$ has integer coefficients
min and max computed using (incremental) LP solver

Coalescing Cases



Coalescing Cases



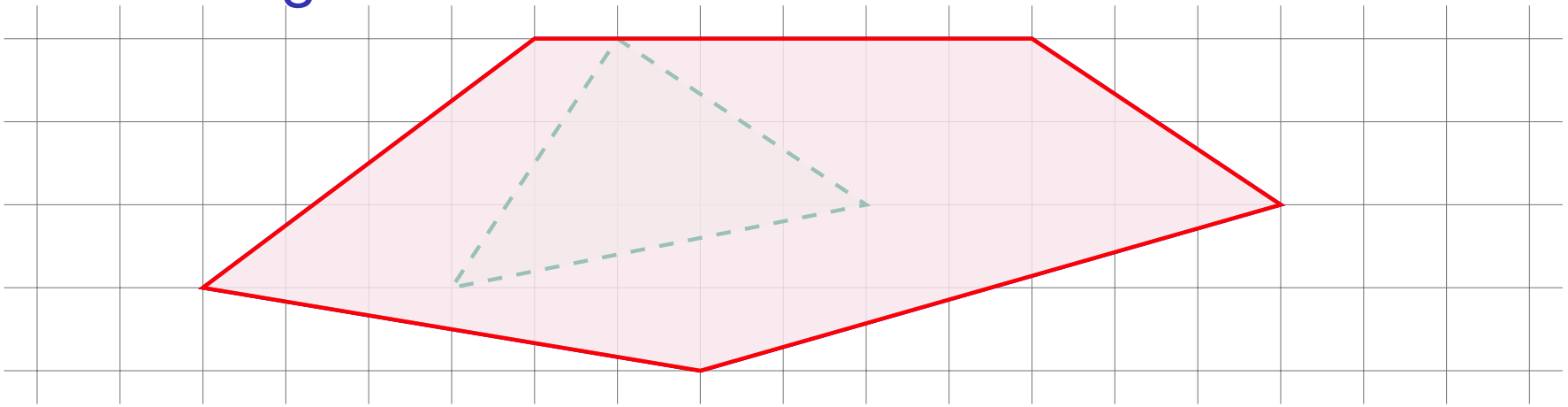
1. All constraints of A are valid for B

   $\Rightarrow$ drop B

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



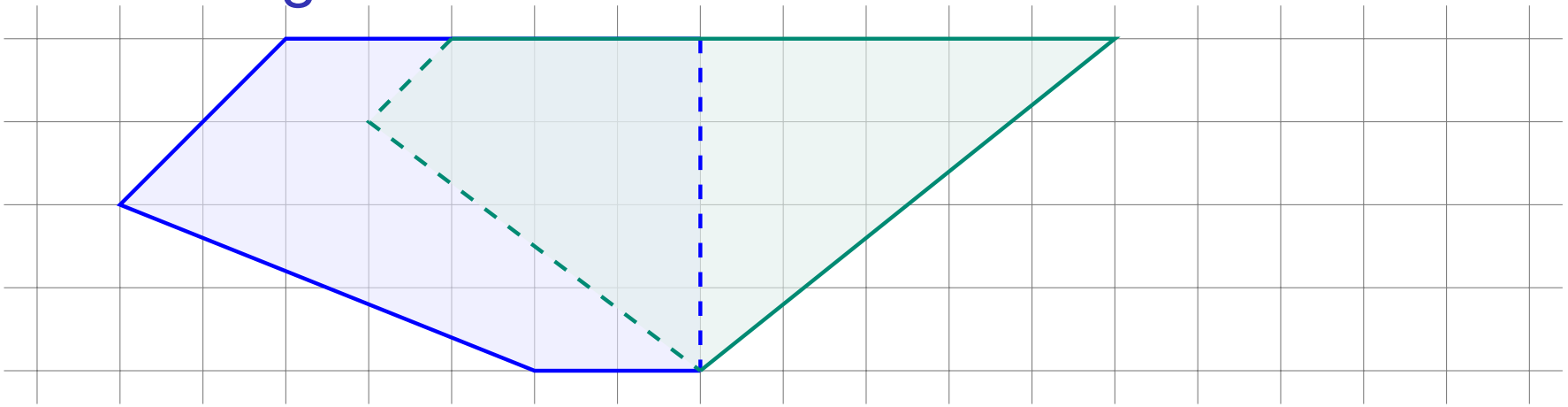
1. All constraints of A are valid for B

   $\Rightarrow$ drop B

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



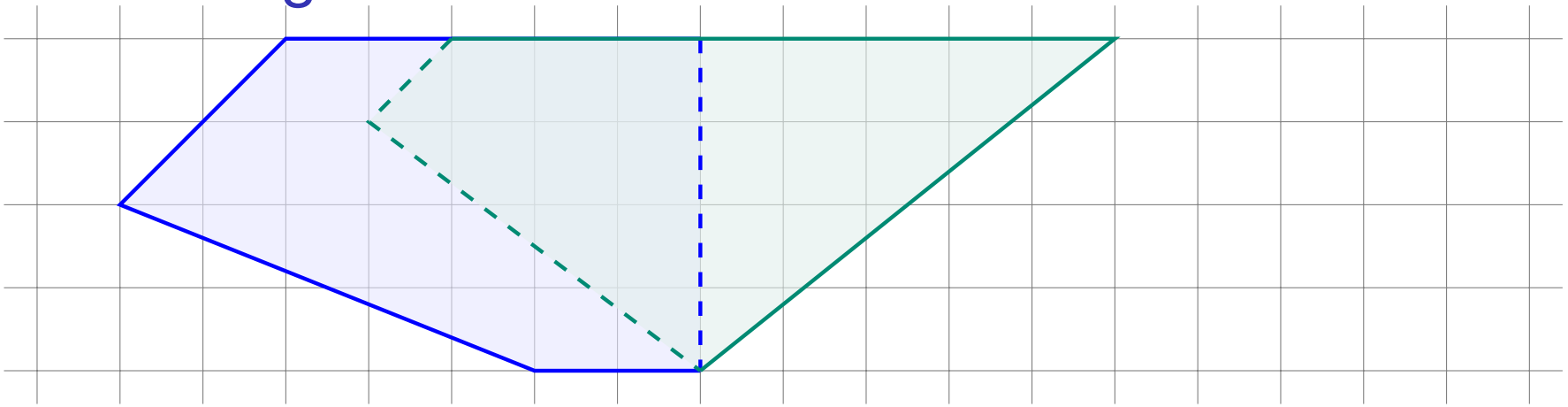
1. All constraints of A are valid for B

   $\Rightarrow$ drop B

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



1. All constraints of A are valid for B

   $\Rightarrow$ drop B

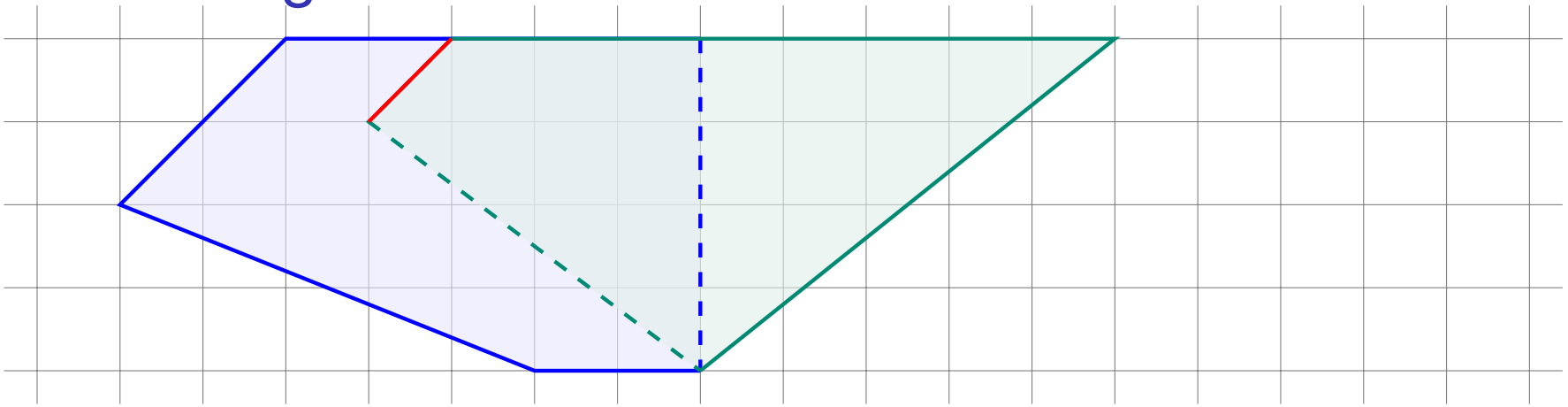
2. Neither A nor B have separating constraints and all cut constraints of A are valid for the cut facets of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



1. All constraints of A are valid for B

   $\Rightarrow$ drop B

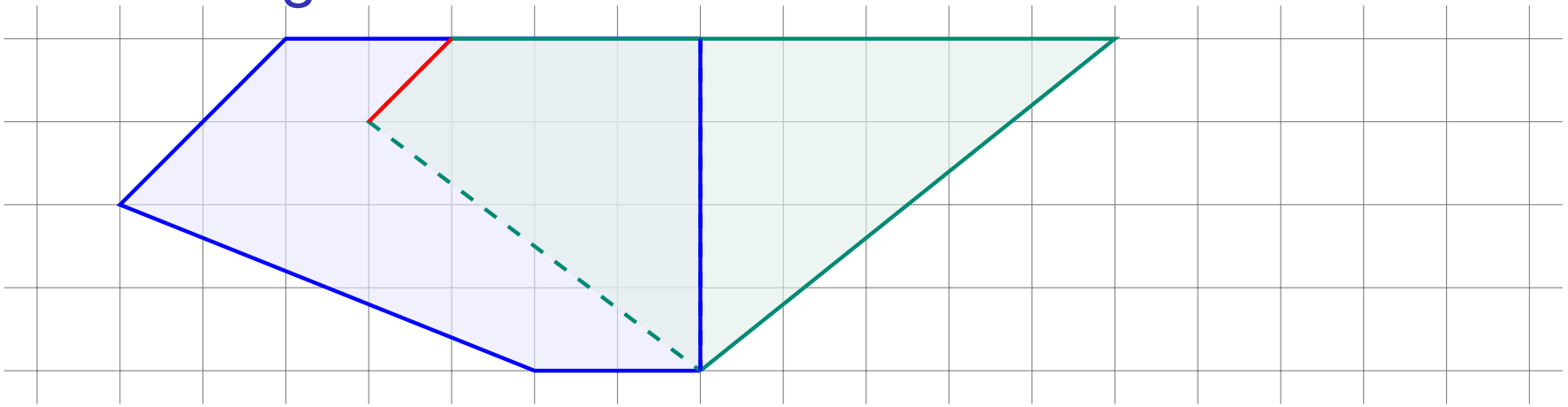
2. Neither A nor B have separating constraints and all cut constraints of A are valid for the cut facets of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



1. All constraints of A are valid for B

   $\Rightarrow$ drop B

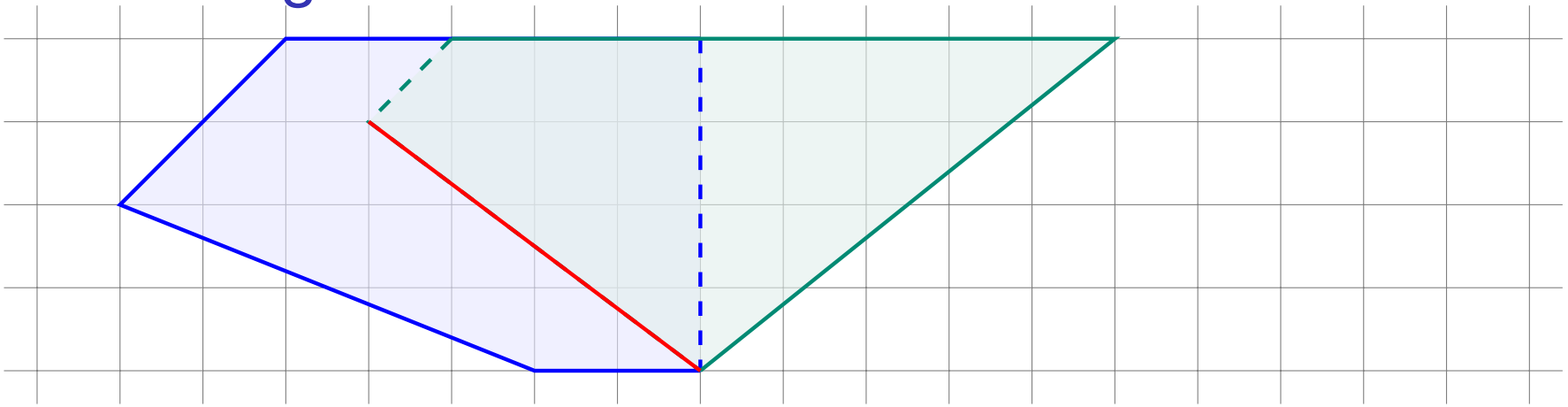
2. Neither A nor B have separating constraints and all cut constraints of A are valid for the cut facets of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



1. All constraints of A are valid for B

   $\Rightarrow$ drop B

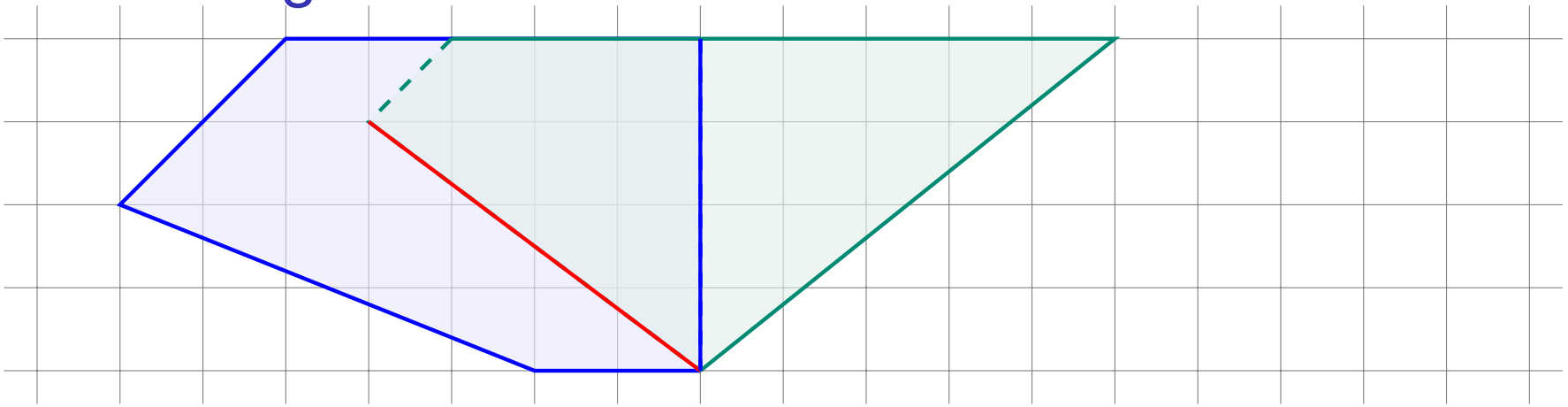
2. Neither A nor B have separating constraints and all cut constraints of A are valid for the cut facets of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



1. All constraints of A are valid for B

   $\Rightarrow$ drop B

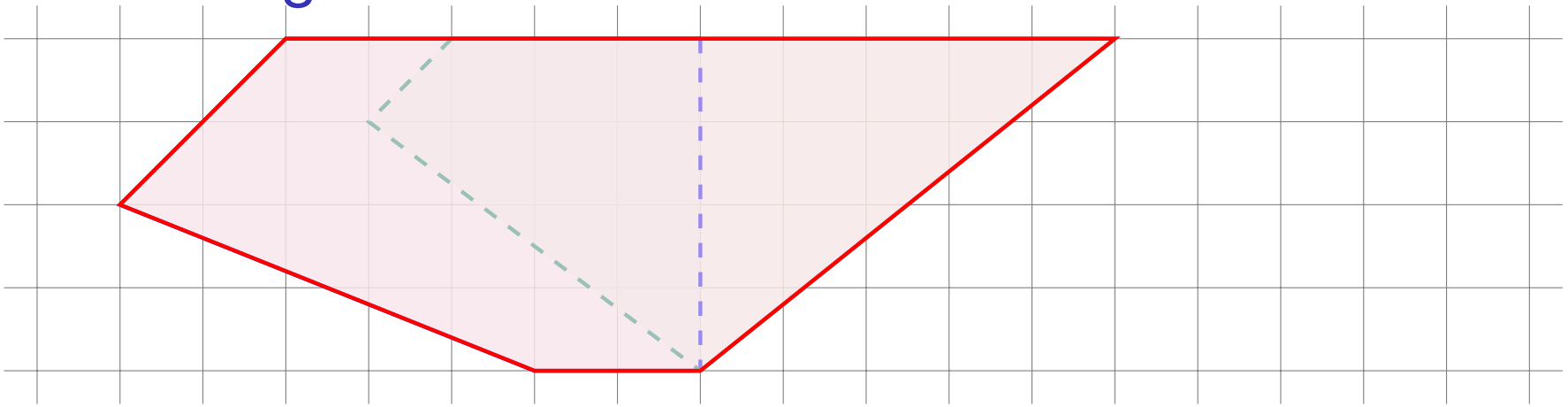
2. Neither A nor B have separating constraints and all cut constraints of A are valid for the cut facets of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



1. All constraints of A are valid for B

   $\Rightarrow$ drop B

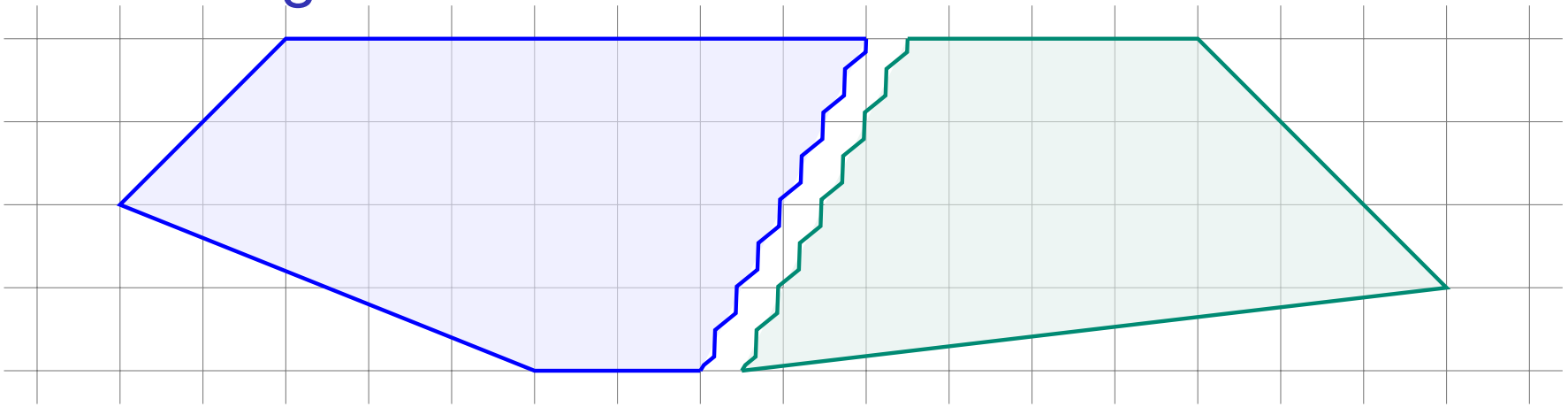
2. Neither A nor B have separating constraints and all cut constraints of A are valid for the cut facets of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



Constraint $t(\mathbf{x}) \geq 0$

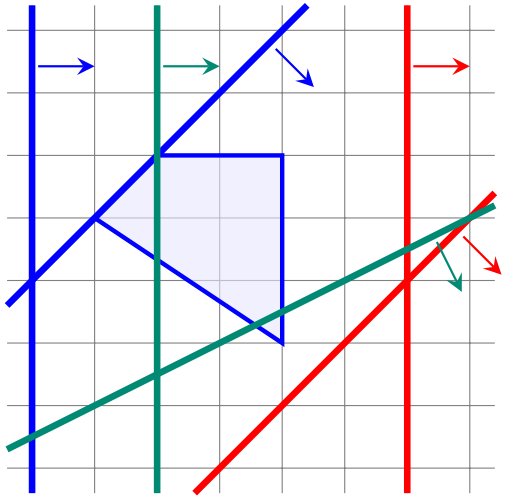
- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Constraint types

Given two disjuncts A and B

For each affine constraint $t(\mathbf{x}) \geq 0$ of A, determine its effect on B

- $\min t(\mathbf{x}) > -1$ over B
  $\Rightarrow$valid constraint
- $\max t(\mathbf{x}) < 0$ over B
  $\Rightarrow$separating constraint



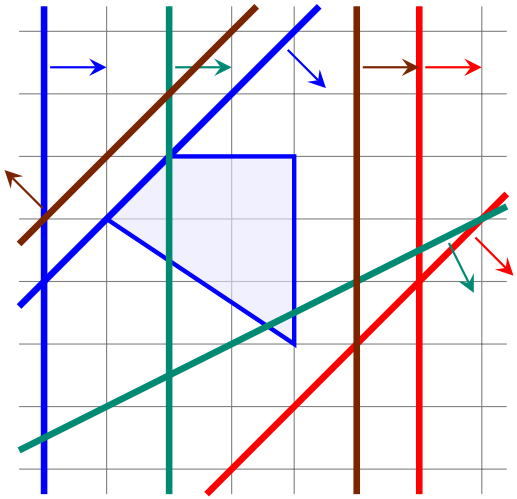
- otherwise (attains both positive and negative values over B)
  $\Rightarrow$cut constraint

Note: affine expression $t(\mathbf{x}) \geq 0$ has integer coefficients
min and max computed using (incremental) LP solver

Constraint types

Given two disjuncts A and B

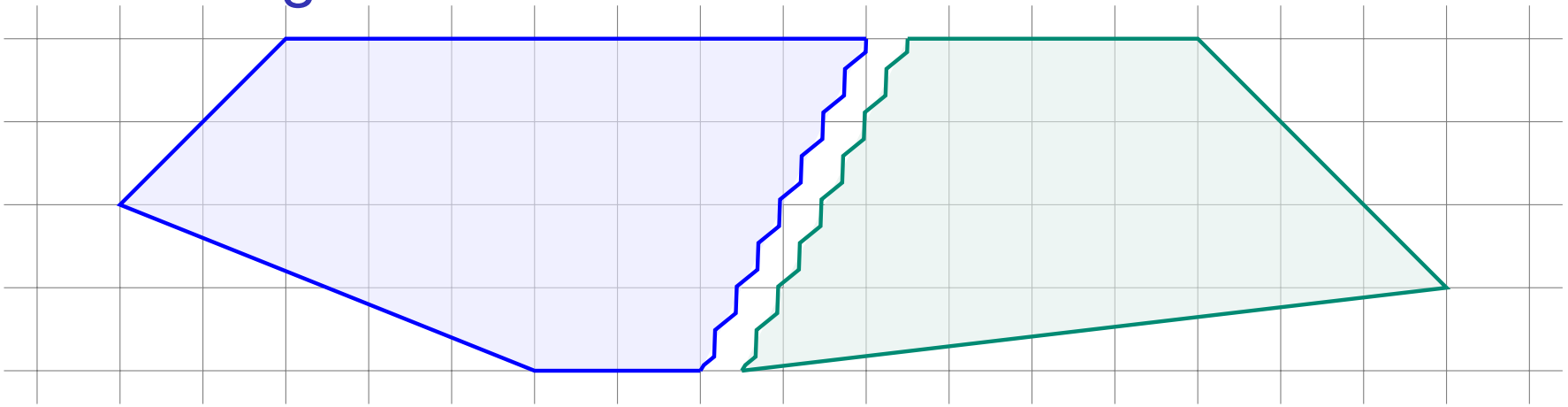
For each affine constraint $t(\mathbf{x}) \geq 0$ of A, determine its effect on B



- $\min t(\mathbf{x}) > -1$ over B
  $\Rightarrow$valid constraint
- $\max t(\mathbf{x}) < 0$ over B
  $\Rightarrow$separating constraint
  special cases:
  - $t = -u - 1$ with $u(\mathbf{x}) \geq 0$ a constraints of B
    $\Rightarrow$ constraint is adjacent to an inequality of B
- otherwise (attains both positive and negative values over B)
  $\Rightarrow$cut constraint

Note: affine expression $t(\mathbf{x}) \geq 0$ has integer coefficients
min and max computed using (incremental) LP solver

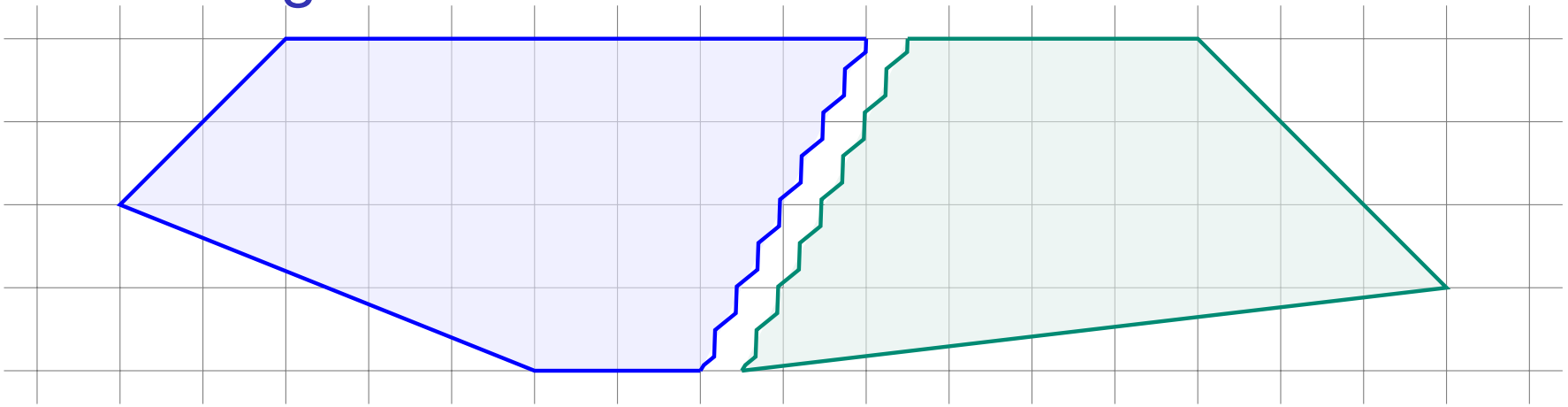
Coalescing Cases



Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
- cut: otherwise

Coalescing Cases



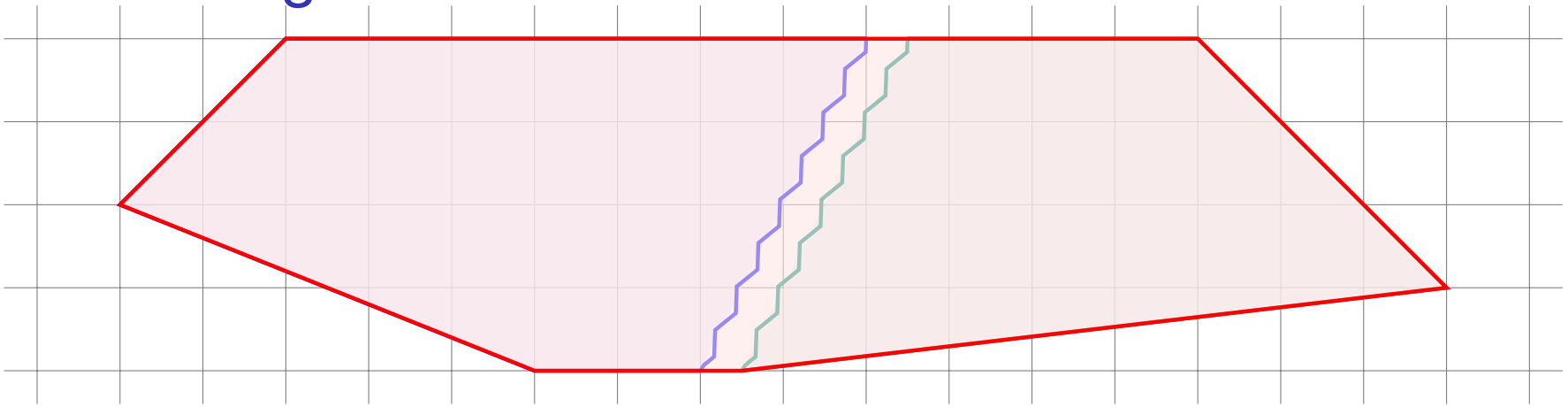
3. single pair of adjacent inequalities (other constraints valid)

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
- cut: otherwise

Coalescing Cases



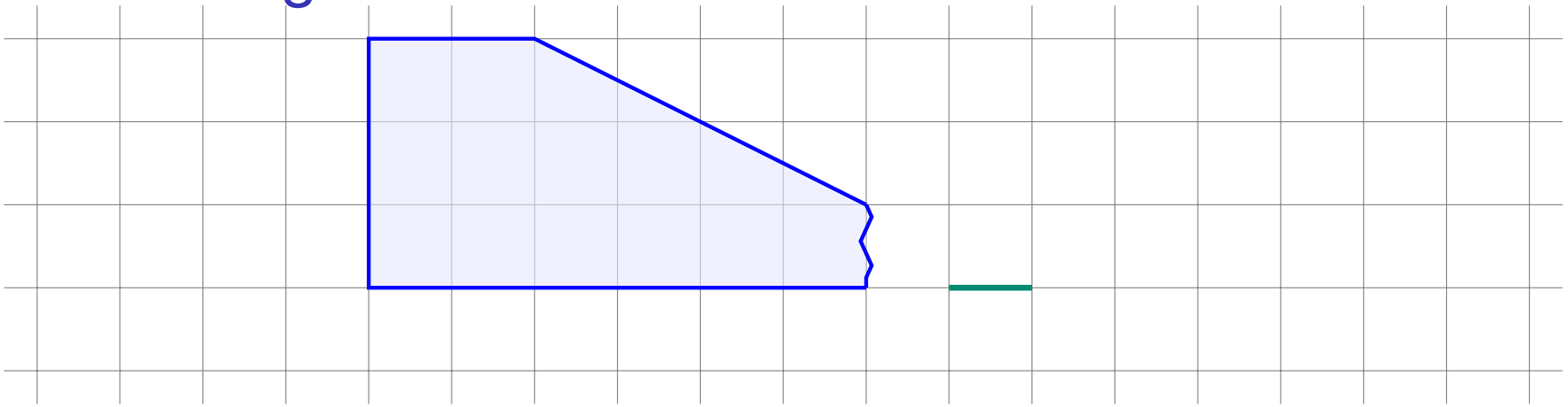
3. single pair of adjacent inequalities (other constraints valid)

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
- cut: otherwise

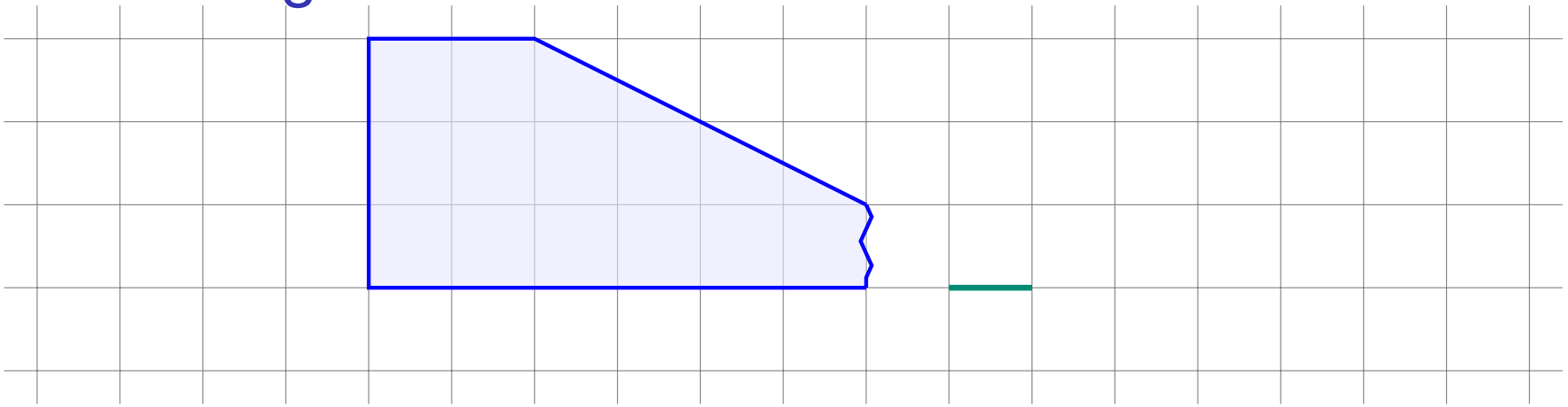
Coalescing Cases



Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
- cut: otherwise

Coalescing Cases

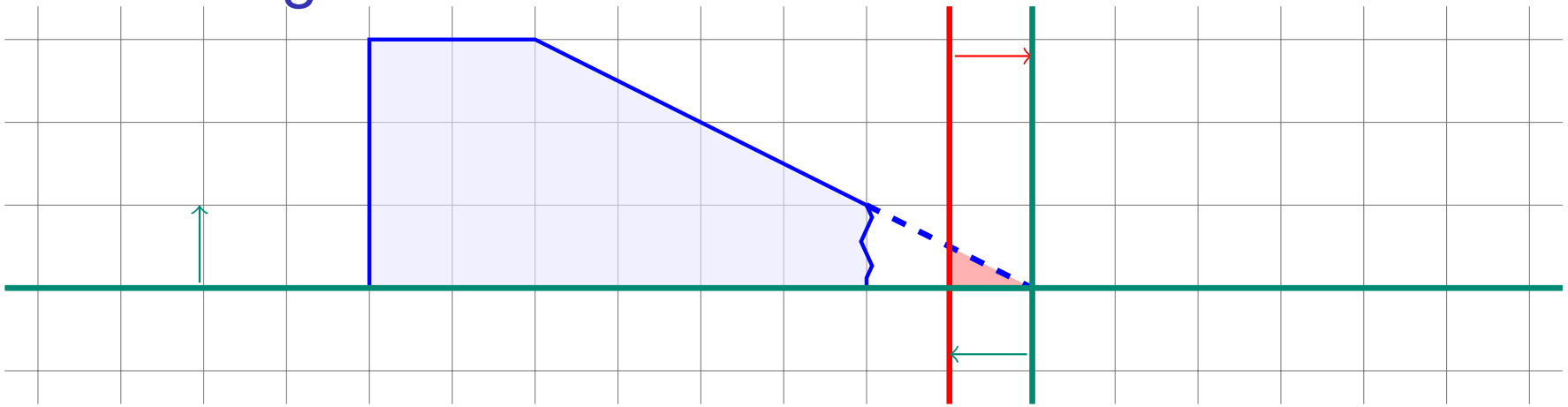


4. A has single inequality adjacent to inequality of B (other constraints of A are valid)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
- cut: otherwise

Coalescing Cases

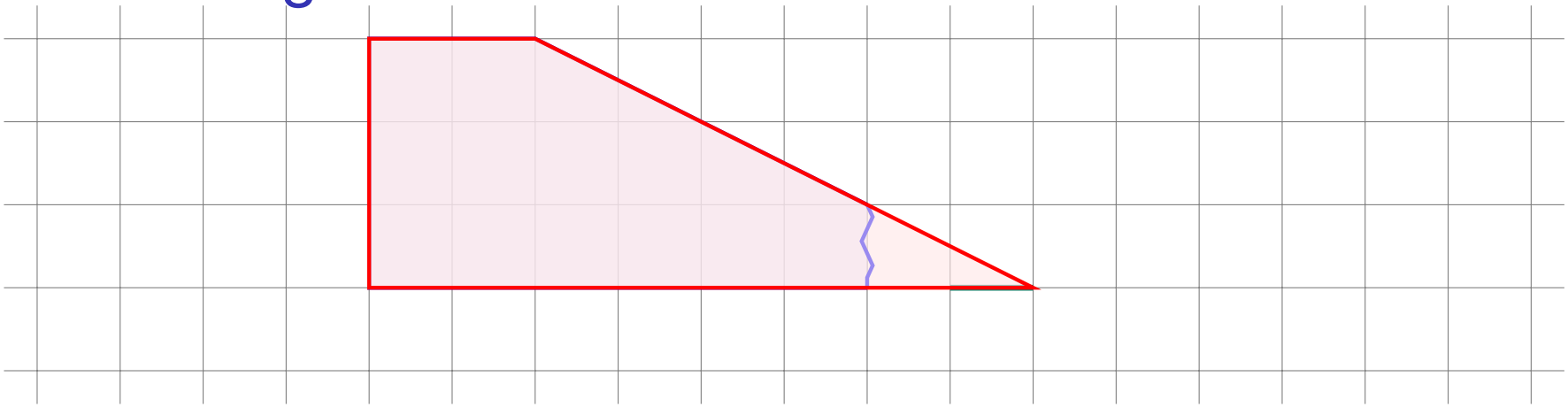


4. A has single inequality adjacent to inequality of B (other constraints of A are valid)
   Result of replacing $t(\mathbf{x}) \geq 0$ by $t(\mathbf{x}) \leq -1$ and adding valid constraints of B is a subset of B
   - $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
- cut: otherwise

Coalescing Cases

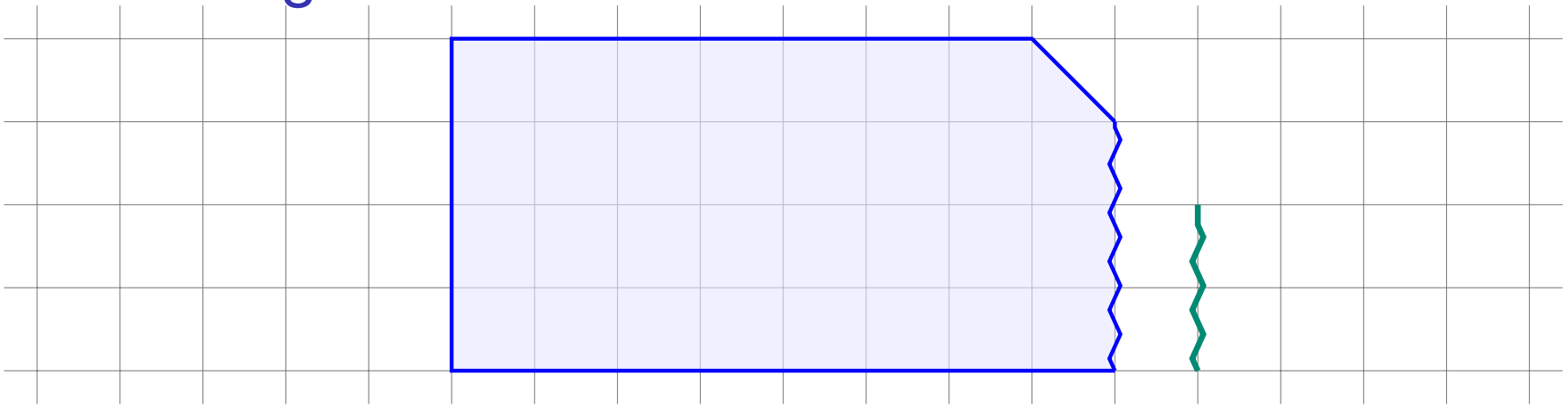


4. A has single inequality adjacent to inequality of B (other constraints of A are valid)
   Result of replacing $t(\mathbf{x}) \geq 0$ by $t(\mathbf{x}) \leq -1$ and adding valid constraints of B is a subset of B
   - $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
- cut: otherwise

Coalescing Cases



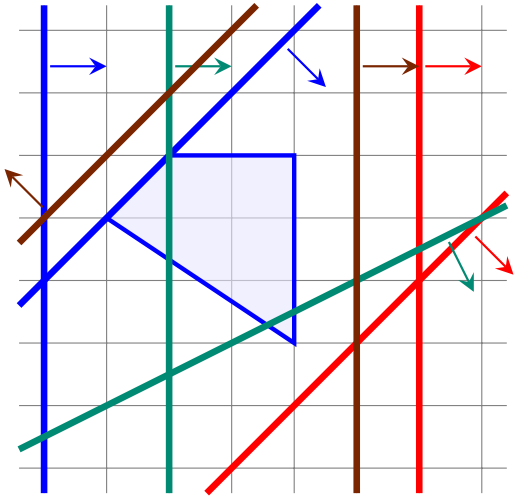
Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
- cut: otherwise

Constraint types

Given two disjuncts A and B

For each affine constraint $t(\mathbf{x}) \geq 0$ of A, determine its effect on B



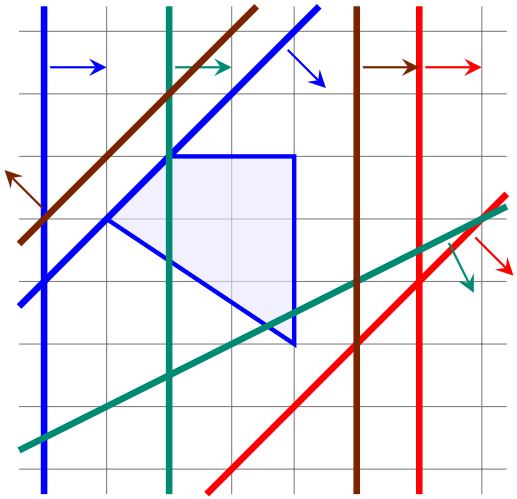
- $\min t(\mathbf{x}) > -1$ over B
  $\Rightarrow$valid constraint
- $\max t(\mathbf{x}) < 0$ over B
  $\Rightarrow$separating constraint
  special cases:
  - $t = -u - 1$ with $u(\mathbf{x}) \geq 0$ a constraints of B
    $\Rightarrow$ constraint is adjacent to an inequality of B
- otherwise (attains both positive and negative values over B)
  $\Rightarrow$cut constraint

Note: affine expression $t(\mathbf{x}) \geq 0$ has integer coefficients
min and max computed using (incremental) LP solver

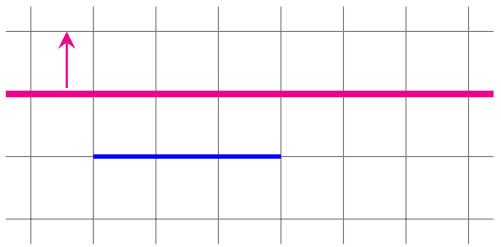
Constraint types

Given two disjuncts A and B

For each affine constraint $t(\mathbf{x}) \geq 0$ of A, determine its effect on B



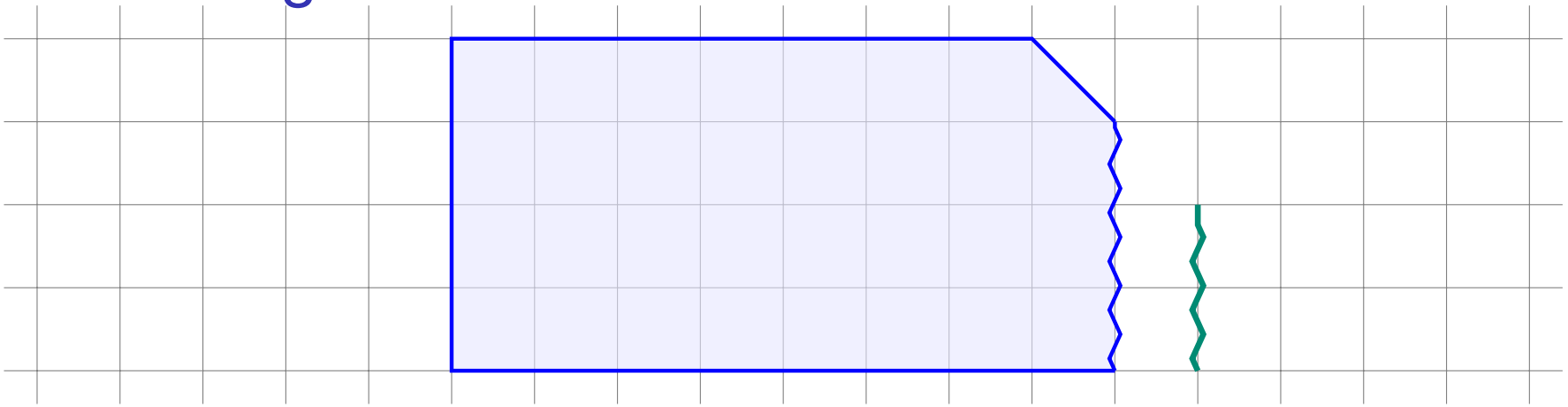
- $\min t(\mathbf{x}) > -1$ over B
  $\Rightarrow$valid constraint
- $\max t(\mathbf{x}) < 0$ over B
  $\Rightarrow$separating constraint
  special cases:
  - $t = -u - 1$ with $u(\mathbf{x}) \geq 0$ a constraints of B
    $\Rightarrow$ constraint is adjacent to an inequality of B
  - $t(\mathbf{x}) = -1$ over B
    $\Rightarrow$ constraint is adjacent to an equality of B



- otherwise (attains both positive and negative values over B)
  $\Rightarrow$cut constraint

Note: affine expression $t(\mathbf{x}) \geq 0$ has integer coefficients
min and max computed using (incremental) LP solver

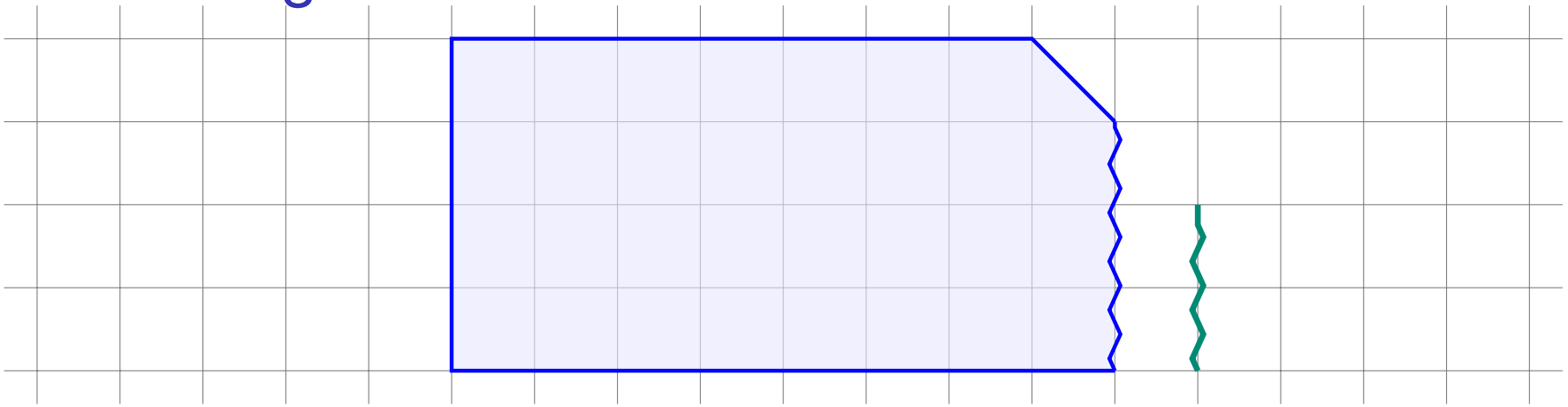
Coalescing Cases



Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
- cut: otherwise

Coalescing Cases

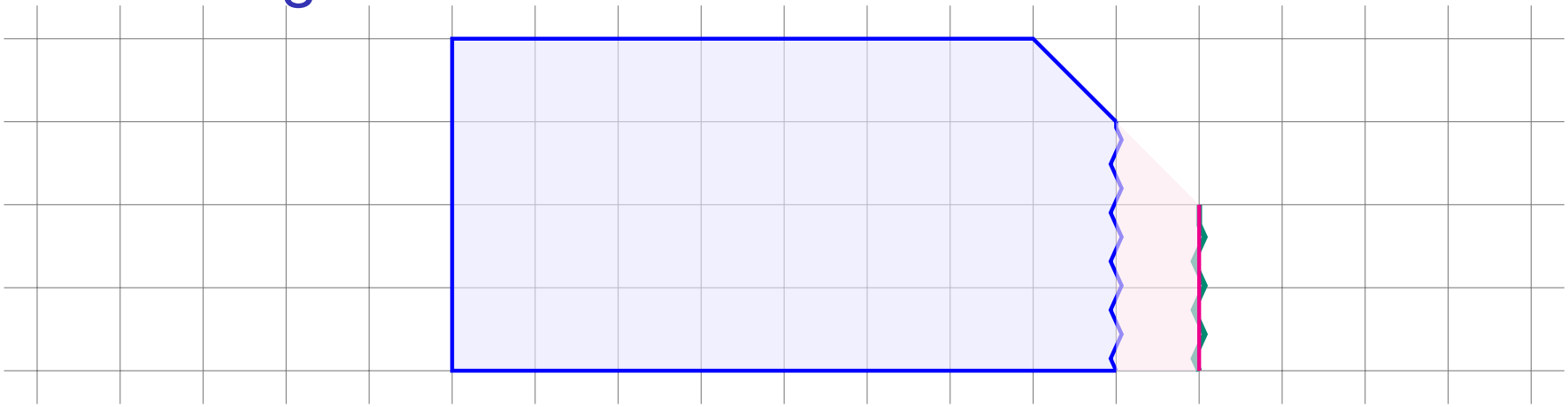


5. A has single inequality adjacent to equality of B (other constraints of A are valid)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



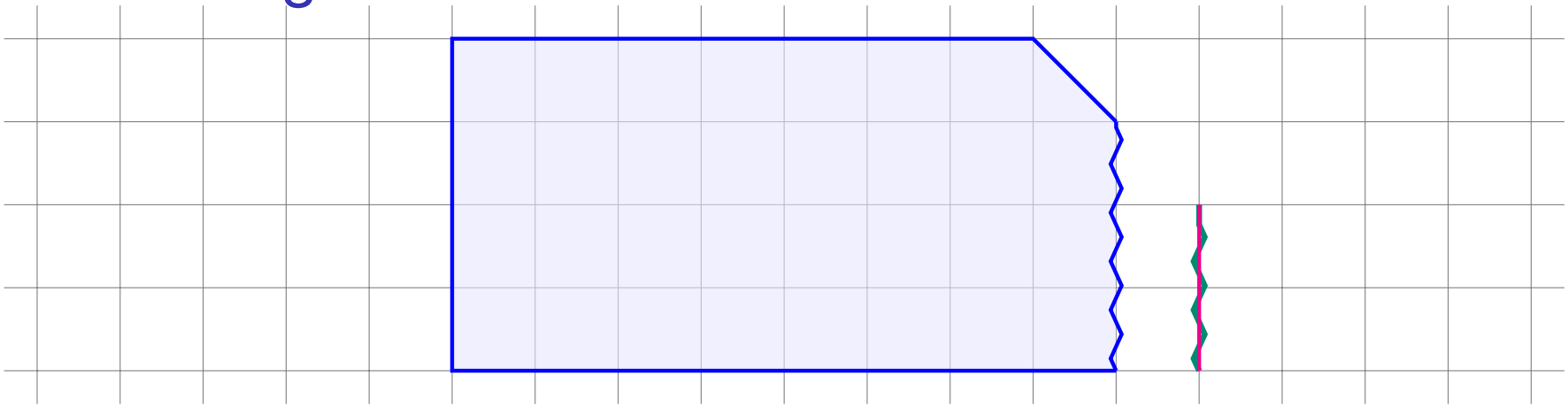
5. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Result of replacing $t(\mathbf{x}) \geq 0$ by $t(\mathbf{x}) \leq -1$ is a subset of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



5. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Result of replacing $t(\mathbf{x}) \geq 0$ by $t(\mathbf{x}) \leq -1$ is a subset of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases

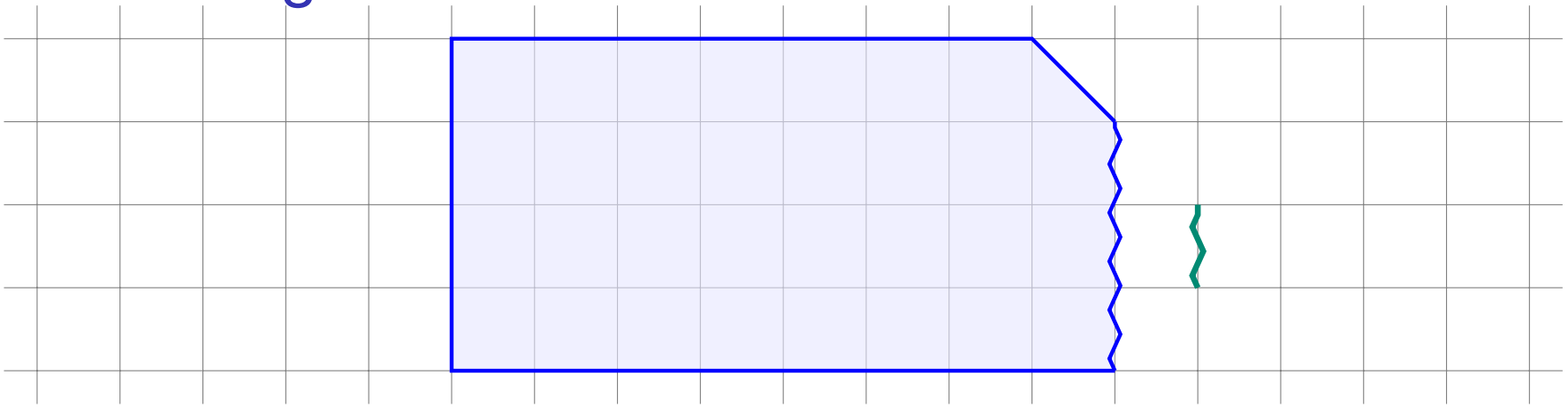
5. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Result of replacing $t(\mathbf{x}) \geq 0$ by $t(\mathbf{x}) \leq -1$ is a subset of B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

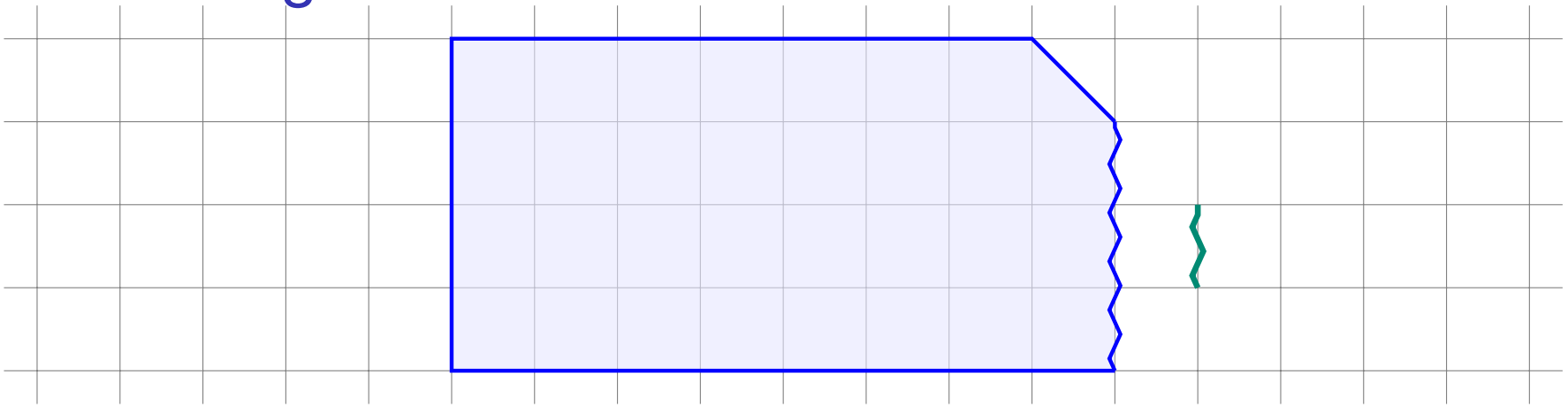
Coalescing Cases



Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



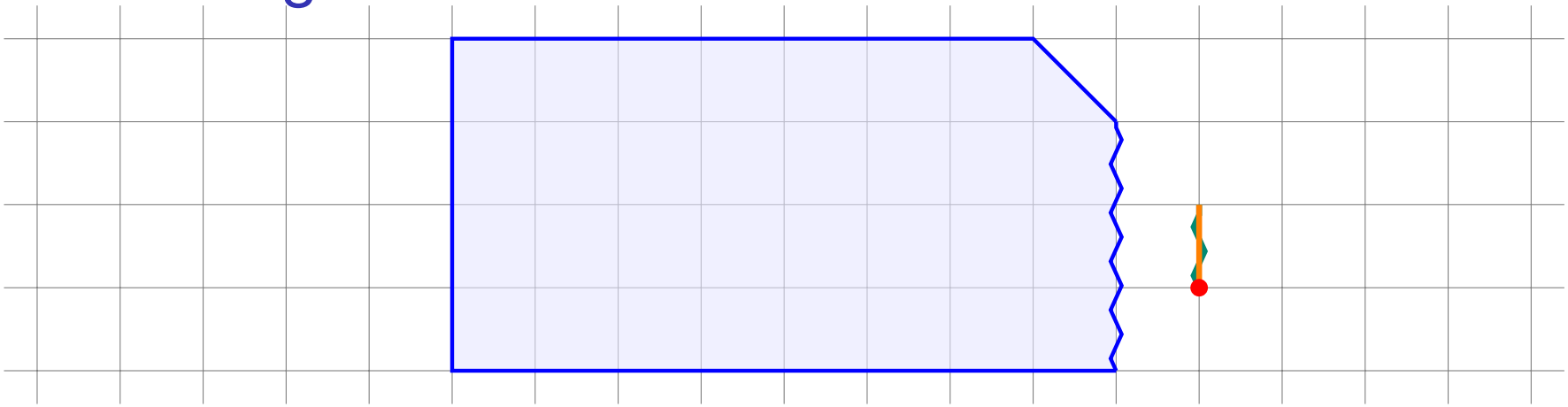
6. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



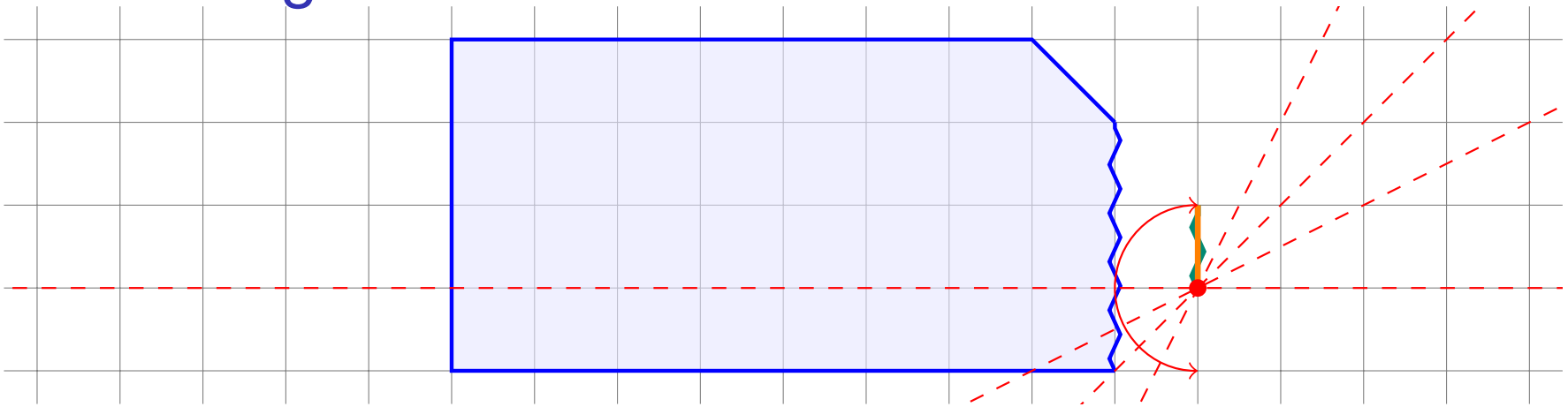
6. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



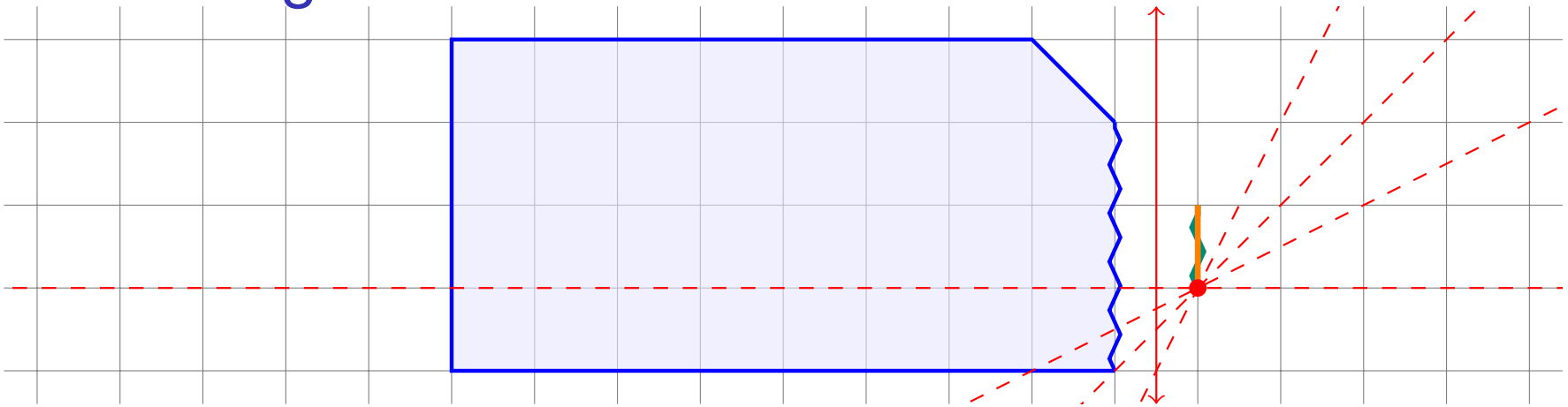
6. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



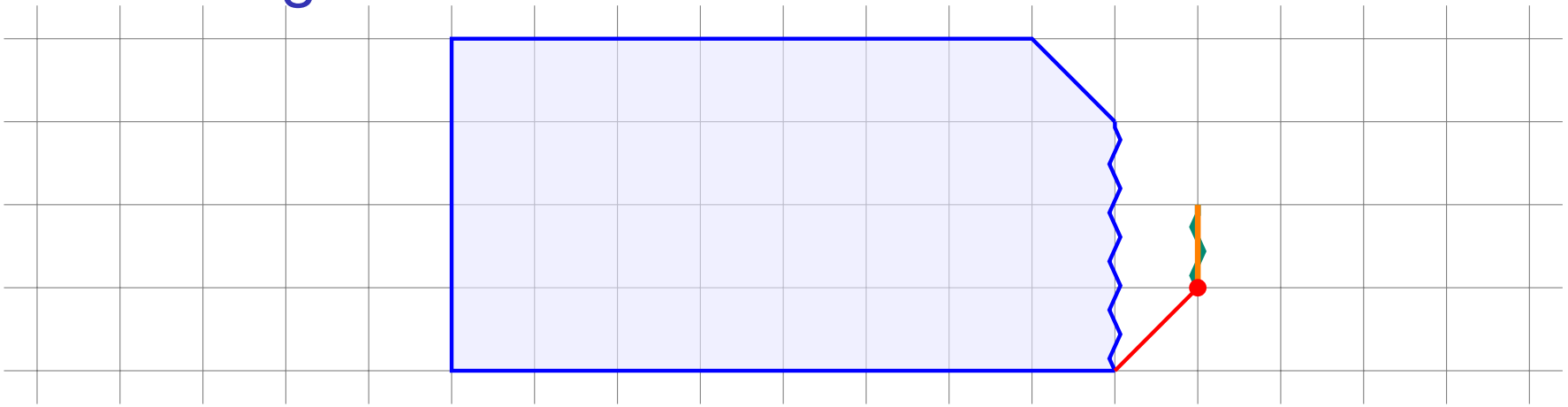
6. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



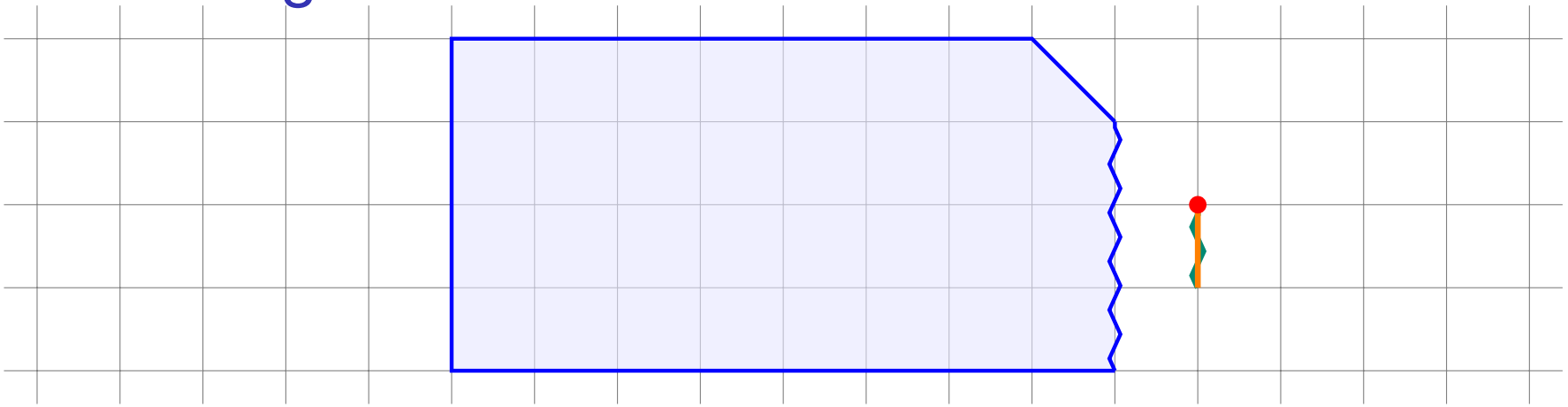
6. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



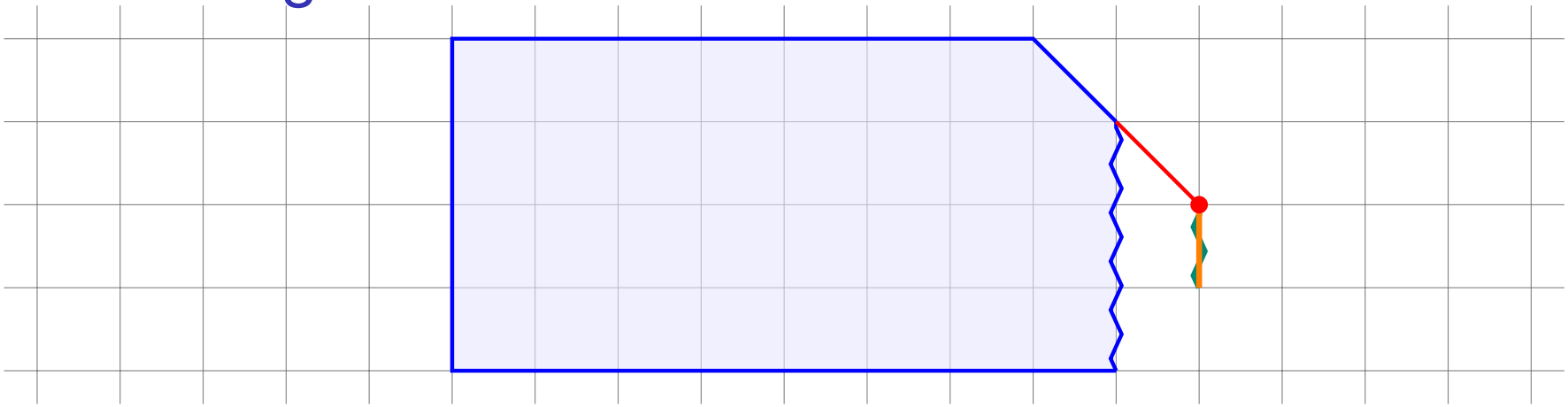
6. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



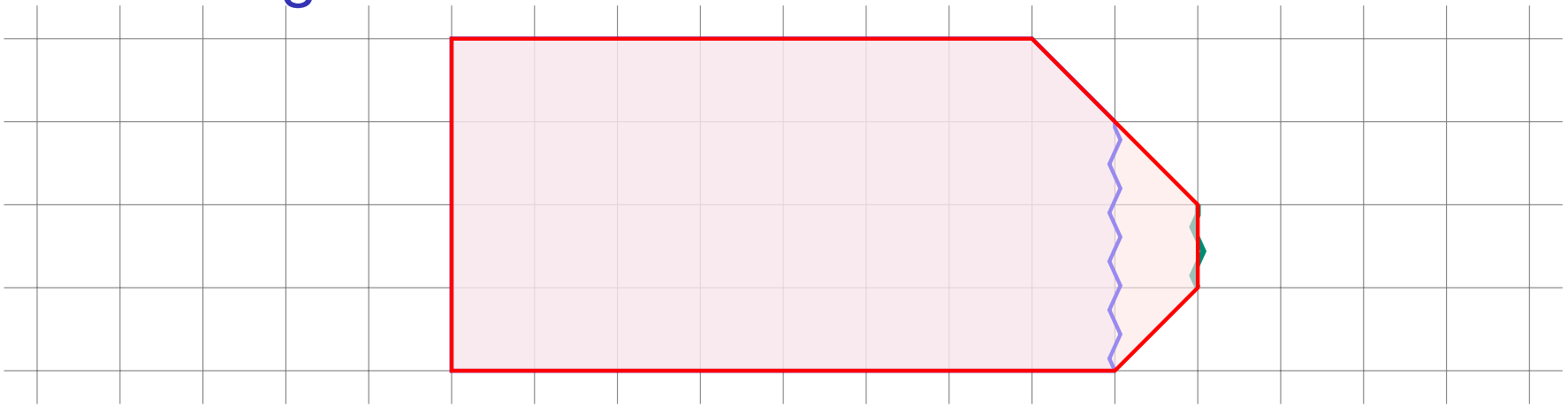
6. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



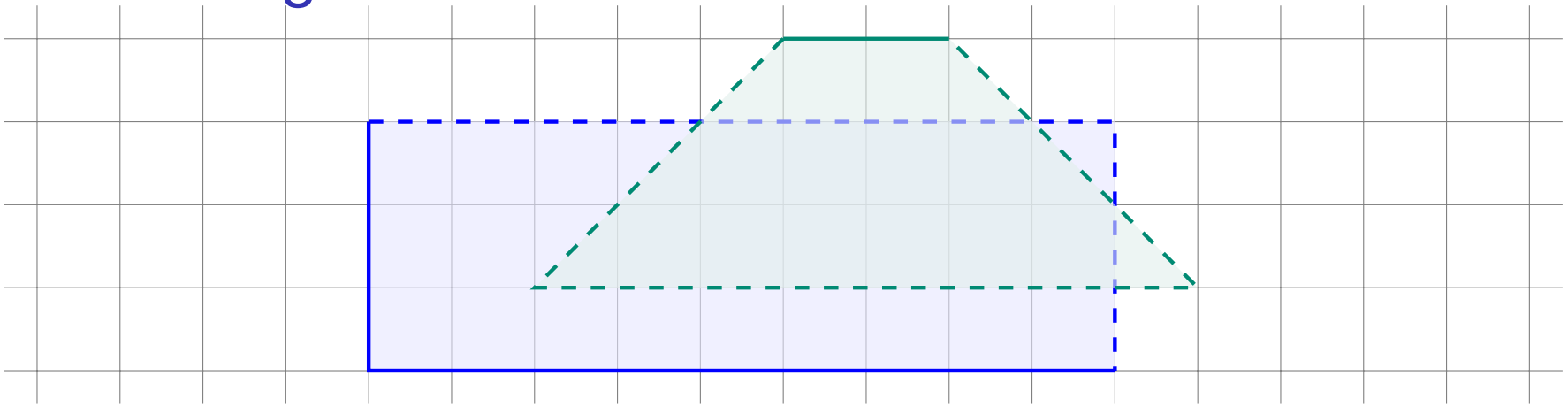
6. A has single inequality adjacent to equality of B (other constraints of A are valid)
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

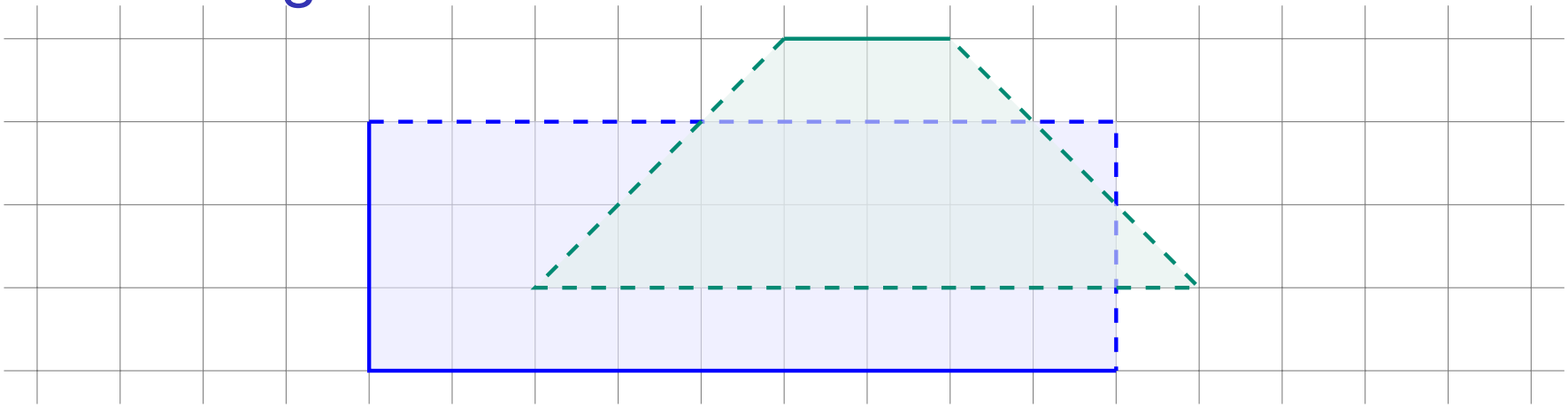
Coalescing Cases



Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



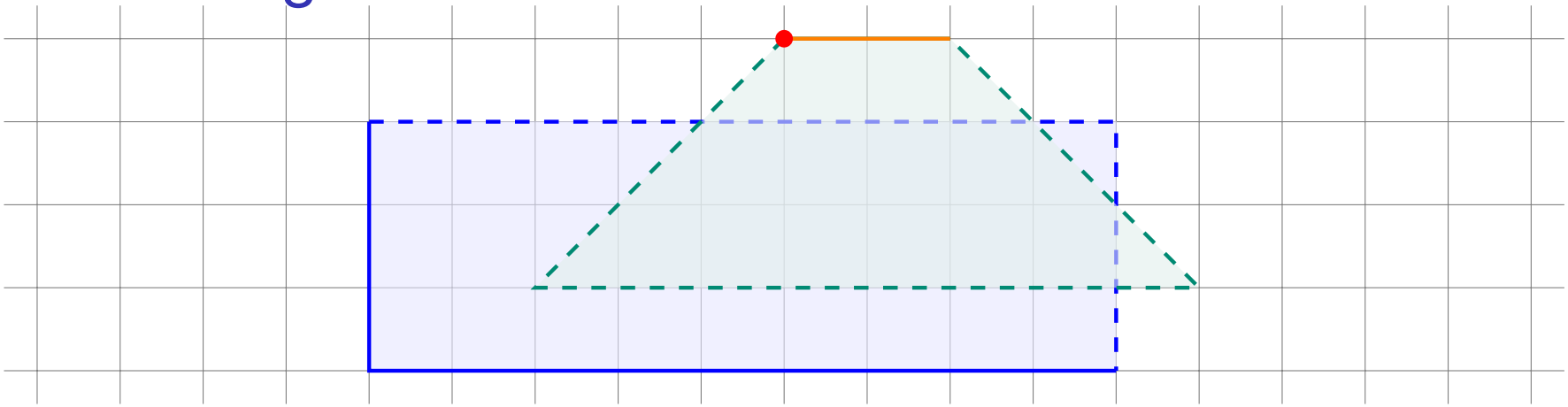
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



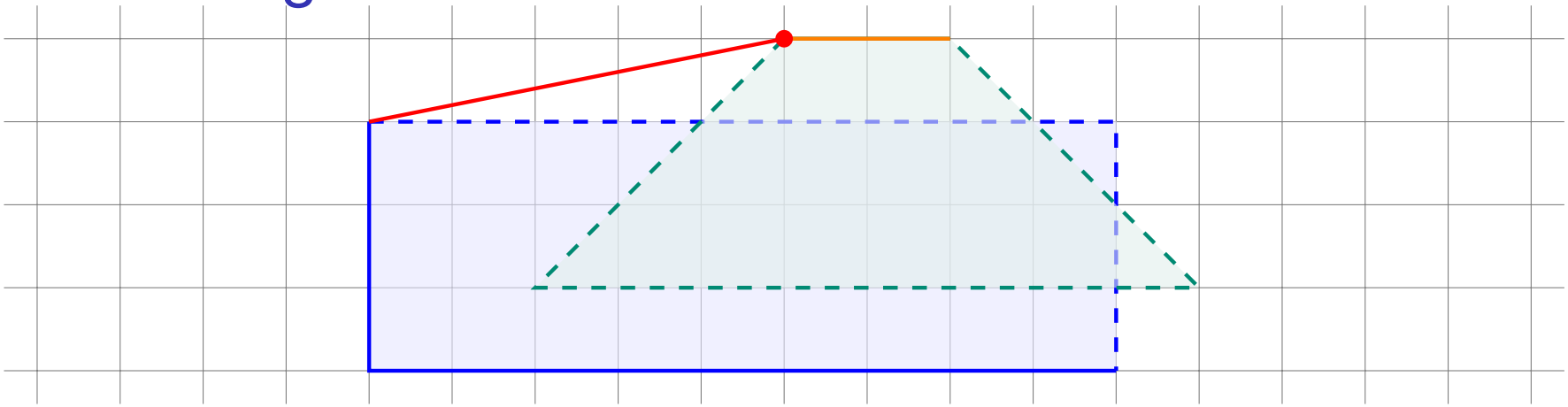
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



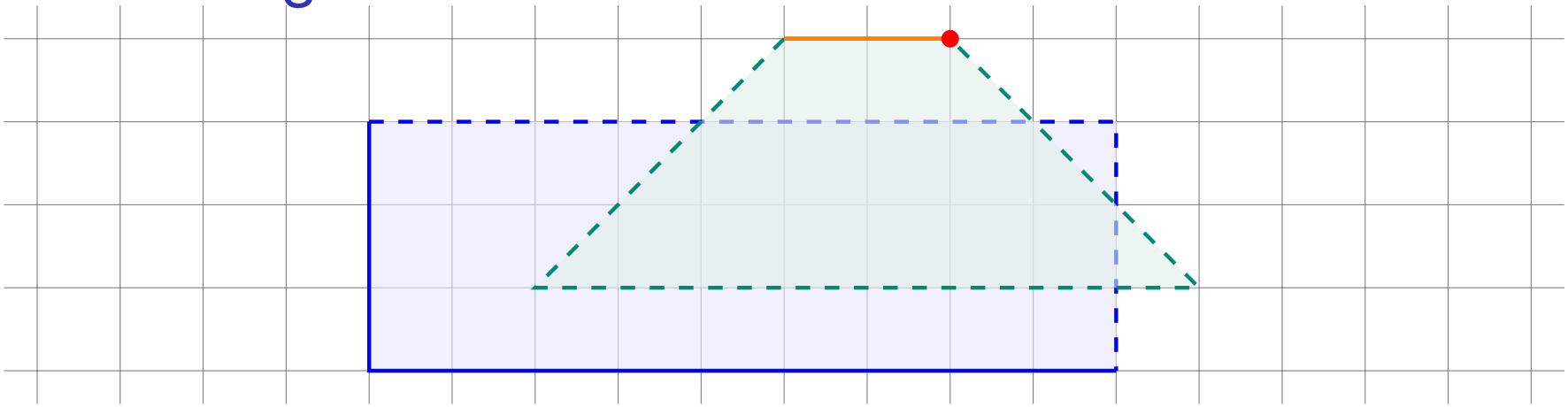
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



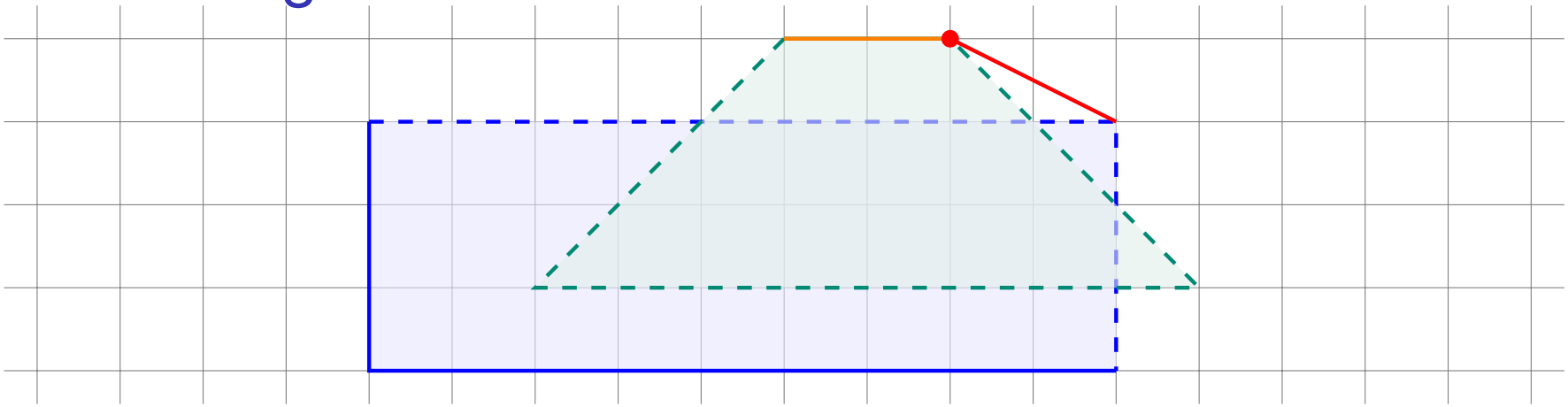
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



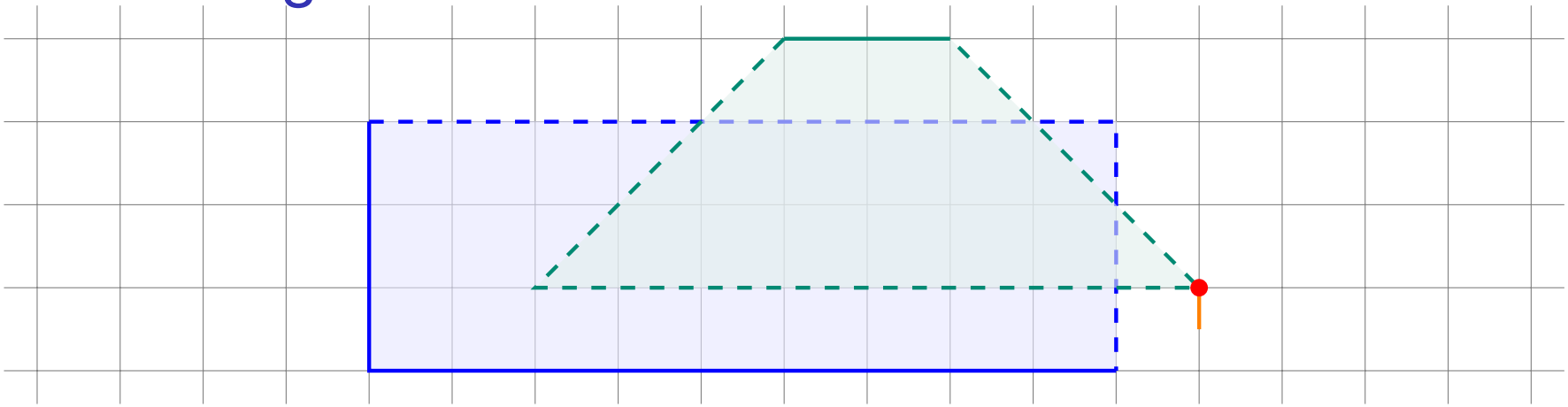
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



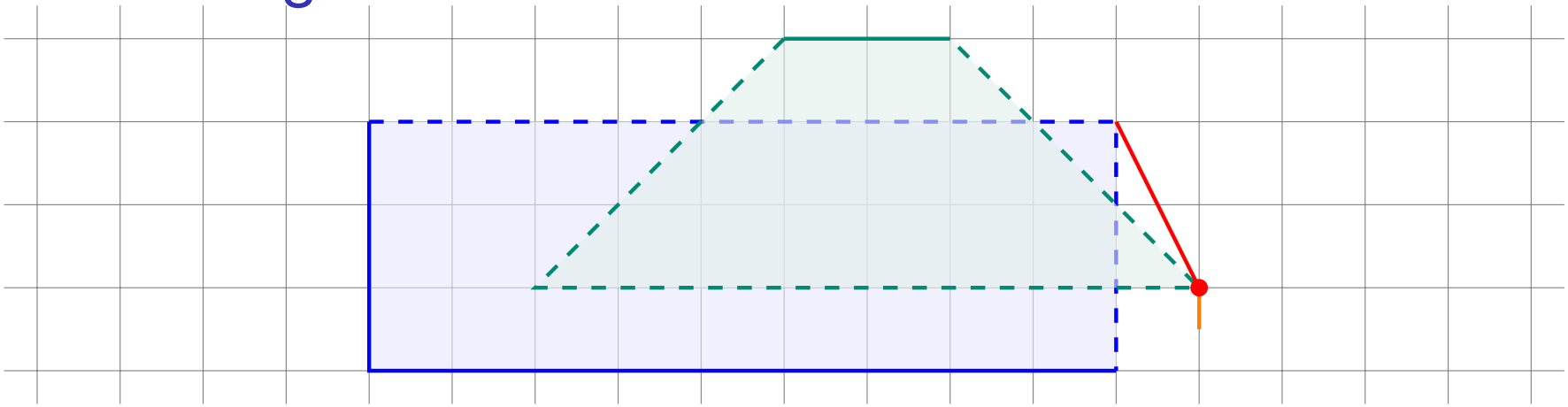
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



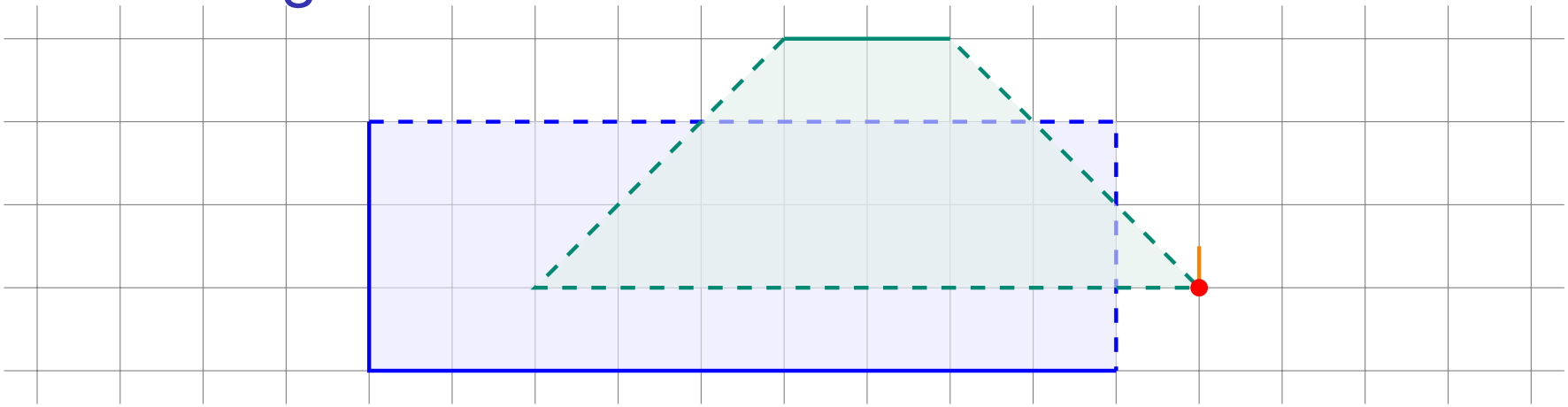
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



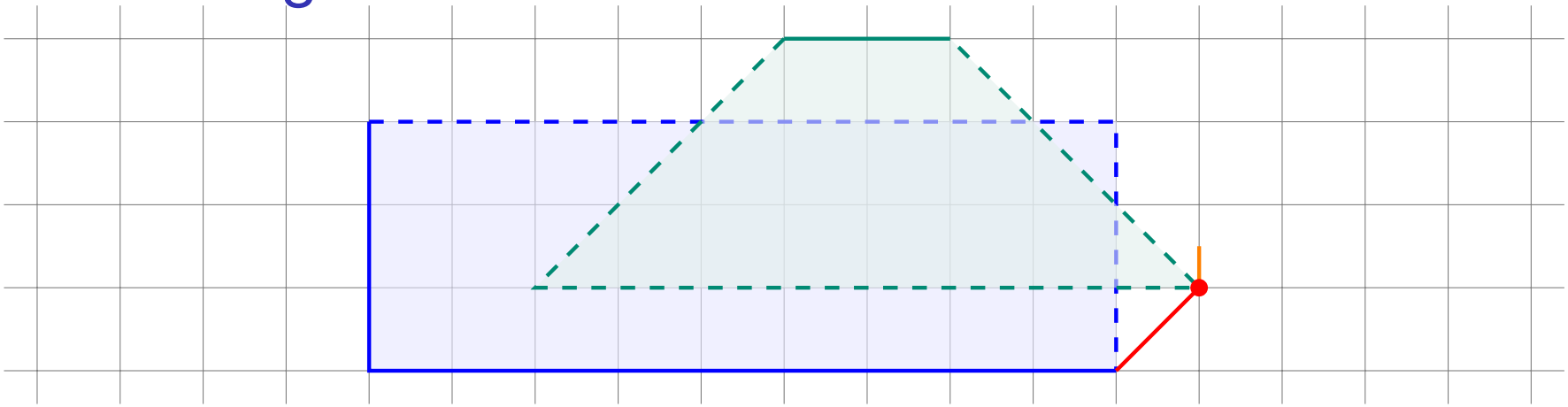
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



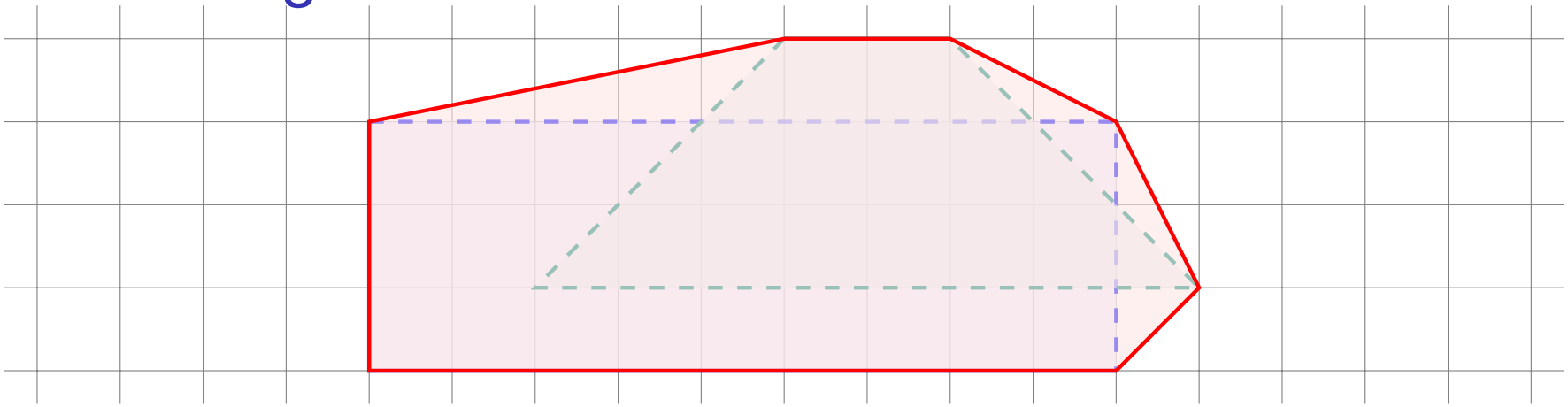
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



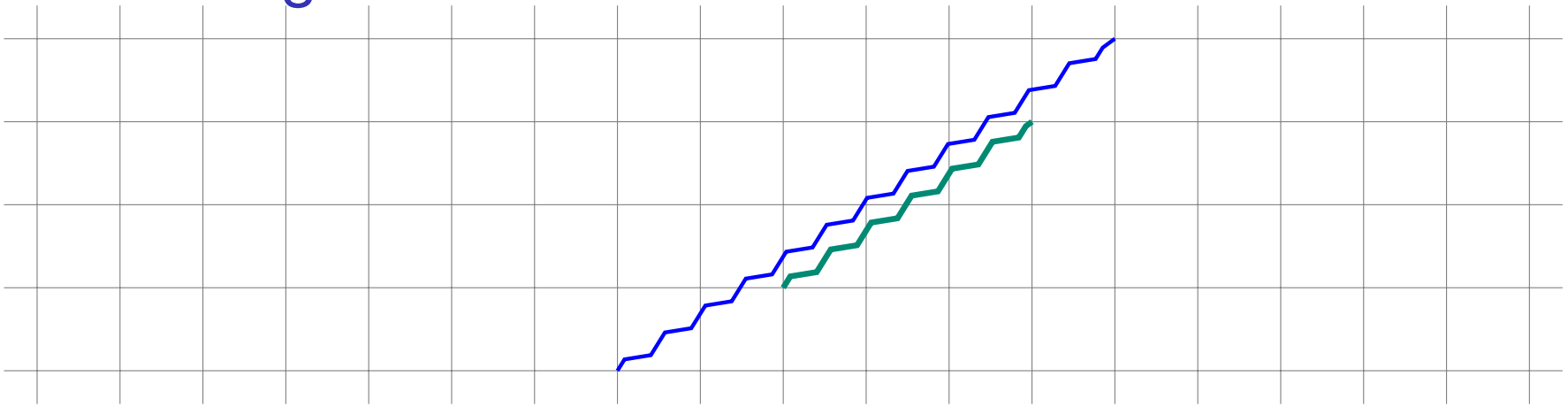
7. B extends beyond A by at most one and all cut constraints of B can be wrapped around shifted facet of A to include A

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints
   (check final number of constraints does not increase)

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

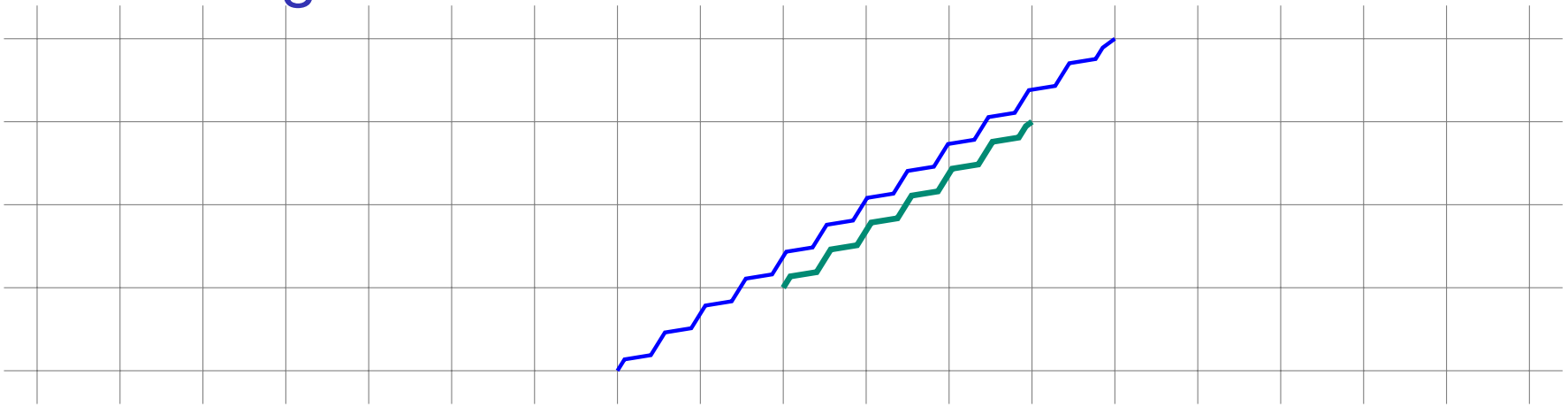
Coalescing Cases



Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



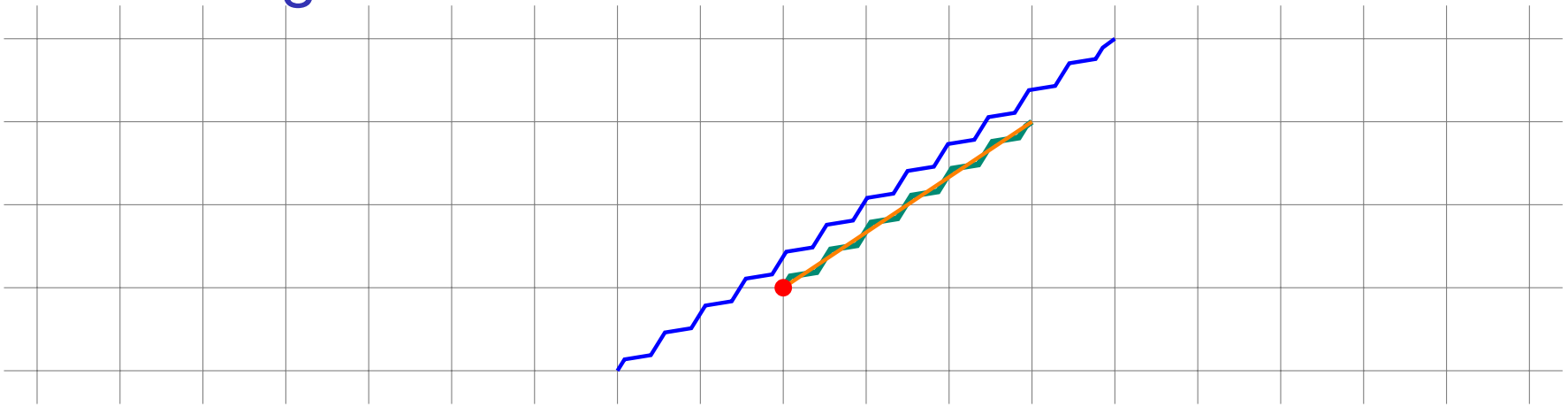
8. A has equality adjacent to equality of B
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A
   Non-valid constraints of A (except $t(\mathbf{x}) \geq 0$) can be wrapped around $t(\mathbf{x}) \leq 0$ to include B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



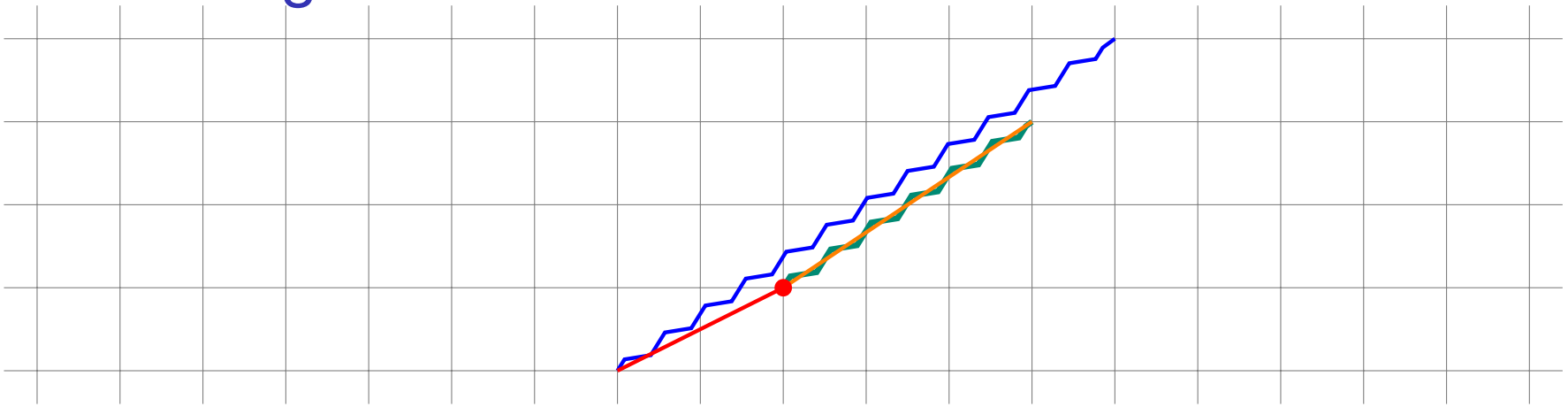
8. A has equality adjacent to equality of B
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A
   Non-valid constraints of A (except $t(\mathbf{x}) \geq 0$) can be wrapped around $t(\mathbf{x}) \leq 0$ to include B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



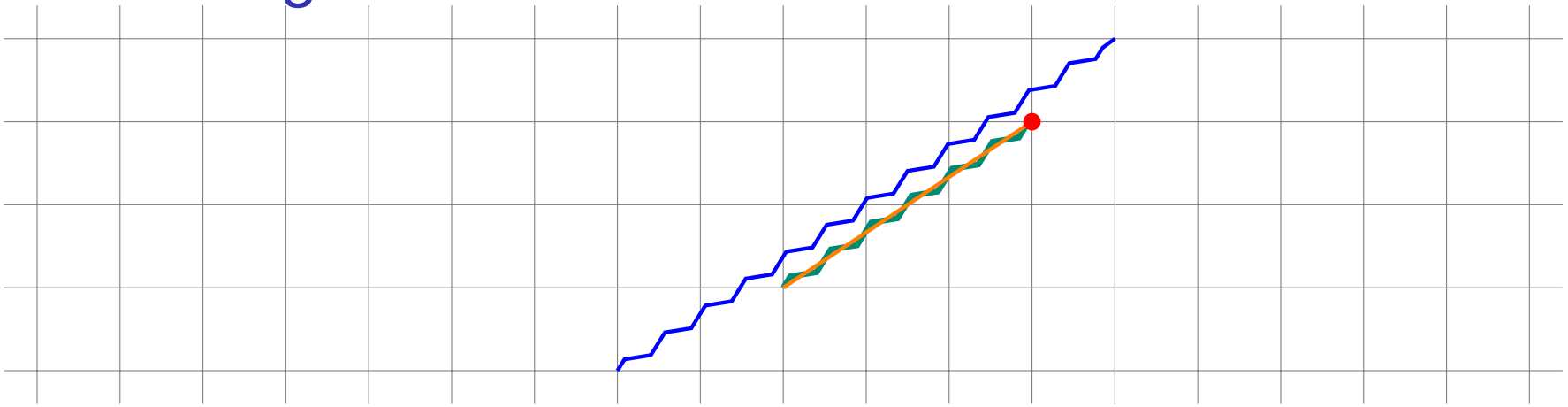
8. A has equality adjacent to equality of B
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A
   Non-valid constraints of A (except $t(\mathbf{x}) \geq 0$) can be wrapped around $t(\mathbf{x}) \leq 0$ to include B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



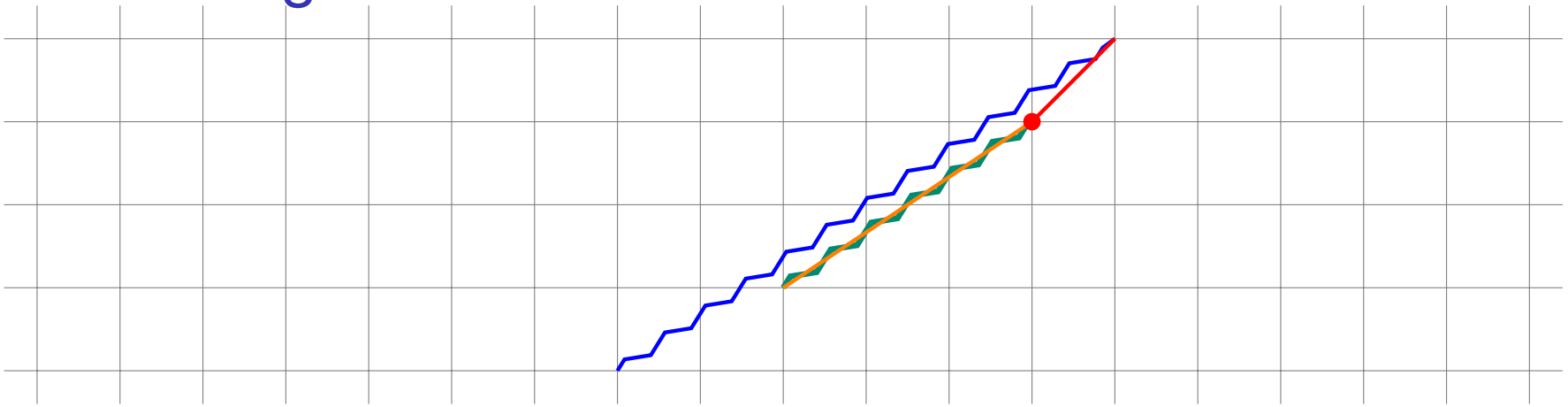
8. A has equality adjacent to equality of B
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A
   Non-valid constraints of A (except $t(\mathbf{x}) \geq 0$) can be wrapped around $t(\mathbf{x}) \leq 0$ to include B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



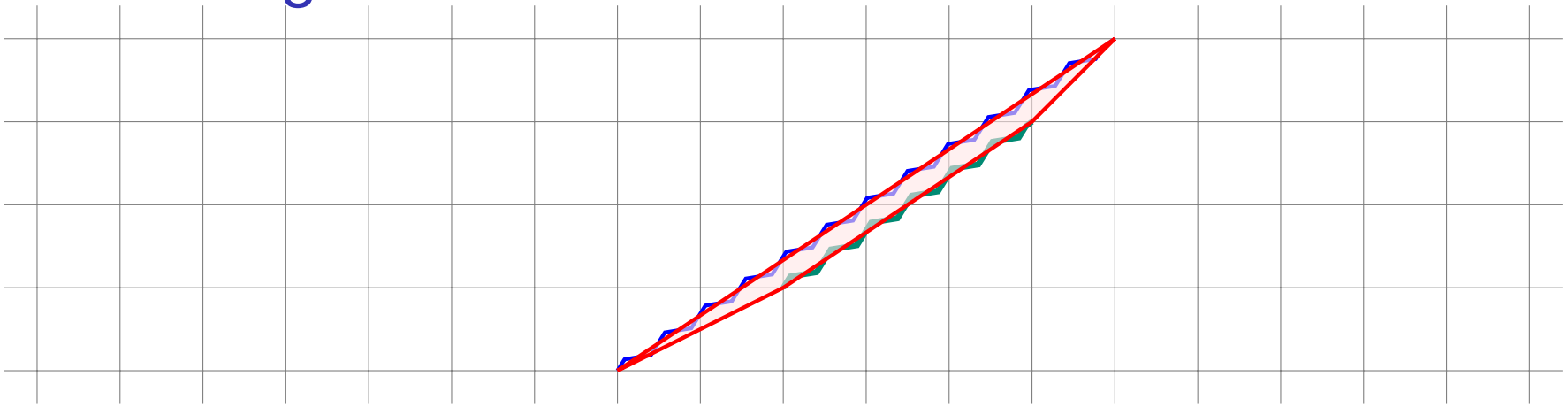
8. A has equality adjacent to equality of B
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A
   Non-valid constraints of A (except $t(\mathbf{x}) \geq 0$) can be wrapped around $t(\mathbf{x}) \leq 0$ to include B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Coalescing Cases



8. A has equality adjacent to equality of B
   Non-valid constraints of B (except $t(\mathbf{x}) \leq -1$) can be wrapped around $t(\mathbf{x}) \geq -1$ to include A
   Non-valid constraints of A (except $t(\mathbf{x}) \geq 0$) can be wrapped around $t(\mathbf{x}) \leq 0$ to include B

   $\Rightarrow$ replace $A \cup B$ by set bounded by all valid constraints and all wrapped constraints

Constraint $t(\mathbf{x}) \geq 0$

- valid: $\min t(\mathbf{x}) > -1$
- separate: $\max t(\mathbf{x}) < 0$
  - adjacent to inequality: $t = -u - 1$
  - adjacent to equality: $t = -1$
- cut: otherwise

Existentially Quantified Variables and Equalities

- Quantifier elimination in `isl` replaces existentially quantified variables by integer divisions of affine expressions in other variables
- These integer divisions are sorted prior to coalescing

- A and B have same number of integer divisions/existentials
  $\Rightarrow$ try all cases
- integer divisions of A form subset of those of B
  (after exploiting equalities of B)
  $\Rightarrow$ check if B is a subset of A
- integer divisions of B form subset of those of A and
  equalities of B simplify away the integer divisions of A not in B
  $\Rightarrow$ introduce integer divisions in B and try all cases

Outline

Conclusions

- it is important to keep the number of disjuncts in a set representation as low as (reasonably) possible
- coalescing in `isl`
  - never increases the total number of constraints
  - based on solving LP problems with same dimension as the original set
  - recognizes a set of patterns